SEE
FOLLOWING
NOTE

VOIR
NOTE
CI-APRÈS

PA[illegible] OF THE ORIGINAL DOCUMENT

ARE OF POOR REPRODUCTIVE QUALITY

CERTAINES PARTIES DU DOCUMENT ORIGINAL

SONT DE PIÈTRE QUALITÉ DE REPRODUCTION

UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

MÉMOIRE D'ÉTUDIANT

CASE POSTALE 6128, STATION A, MONTRÉAL, QUÉBEC, H3C 3J7

METHODES DE RESOLUTION DU PROBLEME D'AFFECTATION DU TRAFIC AVEC DEMANDE VARIABLE

par

Paul Gallant

Université de Montréal

Centre de recherche sur les transports - Publication #131

Avril 1979

Dépôt légal - 4e trimestre 1979

TABLE DES MATIERES

Table des illustrations. iii

Sommaire iv

Introduction 1

Premier chapitre - Modèles mathématiques et aperçu des méthodes de résolution. 5

1.1 Terminologie et notation. 5

1.2 Modèles mathématiques 7

1.2.1 Modèle mathématique obtenu lorsque les inverses des fonctions de demande sont connues 7

1.2.2 Modèles mathématiques obtenus lorsque les inverses des fonctions de demande ne sont pas connues. 10

1.2.2.1 Linéarisation des fonctions de demande. 10

1.2.2.2 Modèle mathématique obtenu en utilisant les fonctions de demande. 16

Deuxième chapitre - Méthodes générales de résolution du problème d'affectation du trafic. 21

2.1 Méthode des directions réalisables. 21

2.2 Décomposition par paires O-D. 22

2.3 Génération progressive des chemins utilisés 24

Troisième chapitre - Une méthode de résolution basée sur l'approximation linéaire progressive des fonctions de demande 27

3.1 Un algorithme d'approximation linéaire. 27

3.2 Amélioration de la méthode. 36

Quatrième chapitre - Algorithmes basés sur la méthode des directions réalisables. 41

4.1 La méthode du gradient réduit généralisé. 41

4.2 Adaptation de l'algorithme GRG au problème d'affectation du trafic 46

4.3 Une direction réalisable. 48

Cinquième chapitre - Une méthode de résolution par décomposition de la fonction objectif et des contraintes. 52

5.1 Introduction. 52

5.2 La méthode de Benders généralisée 52

5.3 Adaptation de la méthode de Benders généralisée 57

5.4 Note sur la résolution du problème à l'étape 2 de l'algorithme B1 66

Sixième chapitre - Analyse des résultats obtenus 67

6.1 Description des programmes. 67

6.1.1 Structure des données et espace-mémoire 67

6.1.2 Description générale des programmes 71

6.2 Résultats obtenus 72

6.2.1 Données utilisées 72

6.2.2 Discussion des résultats. 73

6.2.3 Classification des algorithmes. 77

Conclusion 84

Appendice A1 - Conservation des chemins à l'aide de pivots 86

Appendice A2 - Listes des programmes 94

Remerciements. 168

Bibliographie. 169

TABLE DES ILLUSTRATIONS

Figure 1.1-a	$\int_0^{g_k} U_k(x)dx$.	17
Figure 1.1-b	$\int_{\mu_k}^{\infty} D_k(x)dx + \mu_k D_k(\mu_k)$	17
Figure 3.1	Méthode généralement utilisée pour résoudre (P1) lorsque les inverses ne sont pas connues.	29
Figure 3.2	Itération typique de l'algorithme L1.	37
Figure 3.3	Itération typique de l'algorithme L2.	39
Figure 5.1	Résolution du problème (PP)	62
Tableau 6.1	Programmes utilisés	68
Figure 6.1	Réseau utilisé.	74
Tableau 6.2	Description des arcs et des coûts d'utilisation	75
Tableau 6.3	Description des paires origine-destination et des paramètres des fonctions de demande	76
Tableau 6.4	Comparaison des algorithmes utilisant la méthode de décomposition par paires O-D; critière d'arrêt: coûts des chemins utilisés égaux entre eux à 5% près.	78
Tableau 6.5	Comparaison des algorithmes utilisant la méthode de décomposition par paires O-D; critère d'arrêt: coûts des chemins utilisés égaux entre eux à 1% près.	79
Tableau 6.6	Temps d'exécution de l'algorithme de Frank et Woffe . . .	83
Tableau 6.7	Caractéristiques des approximations linéaires utilisées .	83
Figure A1.1	Arborescence de racine σ_k	89
Figure A1.2	Chemin h.	89
Figure A1.3	T^0.	92
Figure A1.4	T^1.	92

SOMMAIRE

Le problème de la recherche d'un flot d'équilibre dans un réseau routier urbain lorsque la demande de déplacement est variable se formule comme un problème de programmation convexe avec des contraintes linéaires. Cette formulation exige la connaissance à priori des inverses des fonctions de demande. Dans certains cas, il est fort difficile, sinon impossible, d'obtenir ces fonctions inverses.

Ce mémoire est une étude de différentes approches de résolution du problème d'affectation du trafic lorsque la demande est variable et lorsque les inverses des fonctions de demande ne sont pas connues.

Nous avons d'abord analysé l'approche basée sur l'approximation linéaire des fonctions de demande. Nous avons ensuite élaboré une nouvelle formulation du problème qui ne nécessite pas la connaissance des inverses des fonctions de demande, et suggéré plusieurs algorithmes de résolution.

Tous les algorithmes élaborés ont été programmés en Fortran. Les résultats des essais sur l'ordinateur de l'Université de Montréal sont rapportés.

INTRODUCTION

Pour de nombreuses études de planification du transport, il faut souvent prédire la quantité de déplacements qui auront lieu dans un réseau routier, ainsi que les itinéraires choisis par les utilisateurs de ce réseau.

Il existe plusieurs méthodes permettant de résoudre un tel problème. Ces algorithmes recherchent un flot d'équilibre en se basant sur le premier principe de Wardrop [16] qui spécifie que chaque conducteur choisit son itinéraire de façon à minimiser son coût de parcours. L'état d'équilibre dans un réseau de transport est justement caractérisé par le fait qu'aucun automobiliste ne peut diminuer son coût de parcours en changent d'itinéraire.

Plusieurs algorithmes trouvant un tel état d'équilibre solutionnent un problème de programmation convexe à contraintes linéaires qui a été formulé par Bekcmann *et al.* [2,14]. Selon cette formulation, le volume de déplacement entre une origine et une destination données peut être constant ou variable. Dans ce dernier cas le volume de déplacement varie alors selon une certaine fonction de demande décroissante (dont l'argument est le coût de déplacement), et la formulation exige la connaissance à priori de l'inverse de cette fonction. Or, les fonctions de demande sont parfois tellement complexes qu'il devient fort difficile, sinon impossible, de les inverser. A titre d'exemple, considérons les fonctions de demande utilisées dans une étude portant sur le couloir urbain San-Diego / Los Angeles effectuée à l'aide du système "DodoTrans" [19]:

$$w_{ijkp} = c_p \, S_{1i}^{b_{1p}} \, S_{2i}^{b_{2p}} \ldots S_{1j}^{d_{1p}} \, S_{2j}^{d_{2p}} \, (\sum_m I_{ijmp})^{f_p} \, (I_{ijkp} / \sum_m I_{ijmp})$$

où

w_{ijkp} est le nombre de déplacements de la zone i à la zone j en utilisant le mode de transport k et dans l'intention p ;

$S_{\ell i}$, $S_{\ell j}$ sont des facteurs socio-économiques caractérisant les zones i et j ;

$b_{\ell p}$, $d_{\ell p}$, f_p sont des paramètres servant à la calibration de la fonction;

et où

I_{ijkp} est défini comme suit:

$$I_{ijkp} = h_{kp} \, (t_{ijk})^{V_{1kp}} \, (C_{ijp})^{V_{2kp}} \, (w_{ijk})^{V_{3kp}}$$

où

t_{ijk} est le coût de déplacement interzonal entre i et j lorsque l'on utilise le mode de transport k ;

C_{ijp} représente les autres coûts (frais de stationnement, ...);

w_{ijk} est une mesure du temps d'attente;

$V_{\ell kp}$, h_{kp} sont des paramètres de calibration.

Ce mémoire tente donc de répondre à un besoin réel dans le domaine de la planification du transport urbain: il fournit une étude de différentes méthodes de recherche d'un flot d'équilibre dans un réseau de transport qui ne font pas appel explicitement aux inverses des fonctions de demande.

Nous avons d'abord analysé deux approches basées sur la linéarisation des fonctions de demande. La première méthode consiste à inverser une approximation linéaire de la fonction de demande; ces inverses sont ensuite utilisées dans les algorithmes classiques résolvant le problème correspondant à la formulation originelle de Beckmann *et al.* La deuxième approche consiste à linéariser la fonction de demande sur un segment unique contenant la solution optimale, la longueur de ce segment diminuant à chaque étape.

Le deuxième type d'algorithmes que nous avons étudiés se basent sur une nouvelle formulation du problème d'affectation du trafic d'équilibre. Cette nouvelle formulation ne nécessite pas la connaissance des inverses des fonctions de demande; les algorithmes de résolution que nous avons élaborés sont des adaptations de la méthode du gradient réduit généralisé et de celle de Benders; nous avons en plus construit une direction réalisable pour ce nouveau problème.

Le plan de ce mémoire est le suivant. Dans le premier chapitre, nous présentons un modèle mathématique du trafic d'équilibre sans faire appel aux inverses des fonctions de demande. Nous établissons l'équivalence entre les solutions obtenues avec ce modèle et celles obtenues avec le modèle de Beckmann *et al.* Nous exposons aussi les fondements théoriques permettant une linéarisation efficace des fonctions de demande.

Le deuxième chapitre contient une description sommaire de certaines méthodes utilisées pour résoudre le problème formulé par Beckmann *et al.*; nous utiliserons aussi ces méthodes pour solutionner le nouveau problème

que nous avons formulé.

Au troisième chapitre, nous présentons une méthode de résolution du modèle de Beckmann *et al.* basée sur l'approximation linéaire progressive des fonctions de demande.

Les quatrième et cinquième chapitres présentent des algorithmes qui solutionnent le nouveau problème; les méthodes présentées au quatrième chapitre sont basées sur l'algorithme des directions réalisables, tandis que dans le cinquième chapitre nous adaptons la méthode de Benders généralisée.

Tous les algorithmes présentés dans ce mémoire ont été programmés en Fortran sur l'ordinateur CDC Cyber 173 du centre de calcul de l'Université de Montréal. Les résultats numériques sont présentés dans le sixième chapitre, et une copie des programmes est insérée en annexe.

PREMIER CHAPITRE

MODÈLES MATHÉMATIQUES ET APERÇU DES MÉTHODES DE RÉSOLUTION

1.1. Terminologie et notation

Soit un réseau de transport représenté par un graphe $G=(N,A)$ formé d'un ensemble de sommets N et d'un ensemble d'arcs A.

Un sommet $i \in N$ peut être soit une zone d'origine, soit une zone de destination (ou les deux) ou bien une intersection de routes. Un arc $ij \in A$ représente un tronçon de route allant de i à j. Un tronçon de route à deux sens est représenté par deux arcs.

A chaque arc ij est associé un coût d'utilisation $C_{ij}(f_{ij})$, où f_{ij} est le nombre de véhicules sur le segment ij (ou le flot sur l'arc ij) et C_{ij} est une fonction continue croissante de f_{ij}. Cette fonction représente le coût d'utilisation de ij tel que perçu par le voyageur en termes de temps, de coût réel (essence, ...), du nombre de feux de circulation rencontrés, de l'état de la route, etc. C'est donc dire que chaque voyageur perçoit un coût différent correspondant à un état de congestion donné sur un même segment. Cependant, afin d'obtenir un modèle relativement simple, on considérera que tous les usagers perçoivent le même coût.

Soient o_i une zone d'origine et d_j une zone de destination. On dit que (o_i, d_j) forme une paire origine-destination, ou paire O/D. Les paires O/D sont numérotées de 1 à n_p ; soit k l'indice correspondant à

la paire (o_i, d_j). On note par g_k le nombre de véhicules partant de o_i et se rendant à d_j et par μ_k le coût associé à un déplacement de o_i à d_j étant donné un flot de g_k. Ces deux variables sont reliées par la fonction de demande de la $k^{ième}$ paire O/D:

$$g_k = D_k(\mu_k)$$

qui est une fonction continue strictement décroissante. L'inverse de D_k est notée U_k.

Selon le premier principe de Wardrop [16], la $k^{ième}$ paire O/D est en état d'équilibre lorsque tous les automobilistes se rendant de o_i à d_j partagent le même coût de parcours μ_k, indépendamment des chemins qu'ils utilisent, et que ce coût est inférieur ou égal à celui encouru par un voyageur qui emprunterait un chemin non-utilisé. On dit que le réseau a atteint l'état d'équilibre lorsque toutes les paires O/D possèdent cette propriété. Les algorithmes que nous présentons dans ce mémoire recherchent un tel état d'équilibre.

La notation utilisée est résumée dans le tableau suivant:

n_p : nombre de paires O/D

f_{ij}: flot sur l'arc ij

h_ℓ : flot sur le chemin ℓ

Φ_k : ensemble des chemins reliant la $k^{ième}$ paire O/D

Φ : $\bigcup_k \Phi_k$

g_k : demande de déplacement interzonal ($k^{ième}$ paire O/D)

μ_k : coût de déplacement interzonal ($k^{ième}$ paire O/D)

1.2. Modèles mathématiques

1.2.1. Modèle mathématique obtenu lorsque les inverses des fonctions de demande sont connues

Il a été démontré [14] que le problème de minimisation suivant donne le flot d'équilibre dans un réseau de transport:

$$\text{Min } F = \sum_{ij\in A} \int_0^{f_{ij}} C_{ij}(x)dx - \sum_{k=1}^{n_p} \int_0^{g_k} U_k(x)dx \qquad (P1)$$

sujet à:

$$\sum_{\ell\in\Phi_k} h_\ell = g_k \qquad (k=1,\ldots,n_p)$$

$$h_\ell \geq 0 \qquad (\ell\in\Phi_k,\ k=1,\ldots,n_p)$$

$$f_{ij} = \sum_{\ell\in\Phi} a_{ij}^{\ell} h_\ell$$

où U_k est l'inverse de la fonction de demande associée à la $k^{\text{ième}}$ paire O/D et $[a_{ij}^{\ell}]$ est la matrice d'incidence arc-chemin.

Le programme non-linéaire P1 est convexe lorsque les fonctions C_{ij} sont croissantes et les inverses U_k décroissantes. Pour voir que la solution optimale de ce problème est un flot d'équilibre, il suffit d'en écrire les conditions d'optimalité de Kuhn-Tucker (ces conditions sont nécessaires et suffisantes puisque P1 est convexe à contraintes linéaires):
il existe

$\Pi_k \geq 0$, $k=1,\ldots n_p$. tels que

$$\sum_{ij\in A} a_{ij}^{\ell} c_{ij} - \Pi_k \geq 0 \qquad (\ell\in\Phi_k,\ k=1,\ldots,n_p) \qquad (1.1)$$

$$\sum_{ij\in A} a_{ij}^{\ell} c_{ij} - \Pi_k = 0 \qquad \text{si } h_\ell > 0\ (\ell\in\Phi_k,\ k=1,\ldots,n_p) \qquad (1.2)$$

$$\mu_k - \Pi_k \leq 0 \qquad (k=1,\ldots,n_p) \tag{1.3}$$

$$\mu_k - \Pi_k = 0 \qquad \text{si } g_k > 0 \ (k=1,\ldots,n_p) \tag{1.4}$$

où $c_{ij} = C_{ij}(f_{ij})$

Les relations (1.2) et (1.4) signifient que tous les chemins utilisés par les usagers se déplaçant entre la $k^{ième}$ paire O/D (i.e. tous les chemins de Φ_k ayant un flot non-nul) exigent le même temps (ou coût) de parcours Π_k égal au coût μ_k donné par la courbe de demande de la paire origine-destination en question; d'autre part, la relation (1.1) précise que tous les chemins reliant la $k^{ième}$ paire O/D (y compris ceux qui ne sont pas utilisés) exigent un temps de parcours supérieur ou égal au temps μ_k mis pour parcourir un chemin utilisé: cette situation correspond à l'état d'équilibre tel que défini par le premier principe de Wardrop [16].

On peut reformuler le problème P1 en considérant le flot sur les arcs plutôt que le flot sur les chemins. Cette approche donne le problème suivant [2]:

$$\text{Min } F = \sum_{ij\in A} \int_0^{f_{ij}} C_{ij}(x)dx - \sum_{k\in S} \sum_{t\in T} \int_0^{g_{kt}} U_{kt}(x)dx \tag{P1'}$$

sujet à

$$\sum_j f_{ij}^k - \sum_j f_{ji}^k = \begin{cases} \sum_t g_{kt} & \text{si } i=k \\ -g_{ki} & \text{si } i\in T \\ 0 & \text{autrement} \end{cases}$$

pour $i\in N$, $k\in S$

$$f_{ij}^{k} \geq 0 \qquad (ij \in A,\ k \in S)$$

$$g_{kt} \geq 0 \qquad (k \in S,\ t \in T)$$

$$f_{ij} = \sum_{k \in S} f_{ij}^{k} \qquad (ij \in A)$$

où S est l'ensemble des sommets d'origine, T est l'ensemble des sommets de destination, U_{kt} est la fonction de demande de la paire O/D (k,t), et g_{kt} est le flot de k à t. On suppose que $S \cap T$ est vide. Cette dernière hypothèse n'est pas contraignante. En effet, s'il existe un sommet $i \in S \cap T$, on peut le remplacer par deux sommets $i' \in S-T$ et $i'' \in T-S$ tel que tout arc de la forme (k,i) est remplacé par un arc (k,i') et tout arc de la forme (i,k) est remplacé par un arc de la forme (i",k).

Une telle formulation a l'avantage de ne pas exiger la connaissance des chemins reliant chacune des paires O/D. Notons aussi que la méthode de Frank et Wolfe permet de solutionner le problème (P1) en ne considérant que les flots sur les arcs [14].

Tous les algorithmes étudiés dans ce mémoire (sauf celui de Frank et Wolfe que nous avons appliqué au problème (P2), section 1.2.2.) traitent le problème (P1) (ou le nouveau problème formulé à la section 1.2.2.2.) en considérant les flots sur les chemins. De tels algorithmes ont le désavantage d'exiger que l'on connaisse la description de chacun des chemins utilisés en termes d'arcs. Or, conserver une telle information peut s'avérer très coûteux. En annexe, nous présentons une méthode de conservation des chemins qui peut, dans certains cas, exiger moins d'espace-mémoire que la représentation usuelle consistant à énumérer la suite d'arcs composant le chemin.

1.2.2. Modèles mathématiques obtenus lorsque les inverses des fonctions de demande ne sont pas connues

Les deux sections suivantes présentent de nouvelles formulations du problème d'affectation du trafic avec demande variable ne faisant pas appel aux inverses des fonctions de demande.

La première approche utilisée consiste à approximer les fonctions de demande par des fonctions linéaires par morceaux et à utiliser les inverses de ces approximations. Le modèle mathématique obtenu est similaire à (P1) et peut être résolu en employant des algorithmes connus (Frank et Wolfe, gradient réduit, etc.).

La seconde formulation présentée ne fait pas appel aux inverses des fonctions de demande. Le problème de programmation mathématique obtenu est un peu plus difficile à résoudre. Les quatrième et cinquième chapitres présentent des algorithmes qui trouvent la solution optimale de ce nouveau problème.

1.2.2.1. Linéarisation des fonctions de demande

Dans cette section, nous présentons un modèle mathématique basé sur des approximations linéaires L_k des fonctions de demande. Le modèle utilise les inverses L_k^{-1} plutôt que les U_k. L'intégrale de L_k^{-1} est notée IL_k. Nous supposerons les fonctions de demande convexes ou concaves.

Geoffrion [7] définit la meilleure approximation linéaire L d'une fonction F sur le domaine X comme étant celle qui minimise

$$\sup_{x \in X} \{F(x) - L(x)\} + \sup_{x \in X} \{L(x) - F(x)\}$$

Il en déduit que la meilleure approximation linéaire en n morceaux A d'une fonction convexe D est celle qui induit la même erreur maximale ε sur chacun des segments et qui est telle que

$$A(x_i) = D(x_i)$$

où les x_i, $i=1,\ldots,n+1$ sont les extrémités des n segments [7,théorème 4]. Nous ne considérerons donc, par la suite, que cette classe d'approximations linéaires.

L'idée de la procédure de linéarisation de Geoffrion est la suivante. Supposons que l'on veuille une approximation linéaire de la fonction convexe D entre ILIM et SLIM qui soit constituée d'un seul segment; le segment approximant le mieux D est celui passant par (ILIM,D(ILIM)) et (SLIM, D(SLIM)) Si l'on veut maintenant une approximation linéaire en deux morceaux, il suffira de trouver le point d'intersection de ces deux segments; soit (P, D(P)) ce point: P sera tel que l'erreur maximale sur le segment [(ILIM,D(ILIM)), (P,D(P))] sera égale à l'erreur maximale sur le segment [(P,D(P)), (SLIM,D(SLIM))]. La procédure récursive suivante trouve la meilleure approximation linéaire en n morceaux de la fonction convexe (ou concave) D sur le domaine [ILIM, SLIM] (les valeurs de n, ILIM et SLIM sont les paramètres d'entrée de la procédure; les paramètres de sortie sont les n-1 points d'intersection) [7]:

Etape 1: Poser une valeur initiale P; définir temporairement [(ILIM,D(ILIM)), (P,D(P))] comme étant le premier segment.

Etape 2: Trouver l'approximation linéaire optimale en n-1 morceaux entre P et SLIM.

Etape 3: Si l'erreur maximale sur [ILIM, P] est égale à l'erreur maximale sur chacun des n-1 segments entre P et SLIM, terminer; sinon, modifier la valeur de P et aller en 2.

On peut utiliser une recherche dichotomique pour déterminer la valeur optimale P* de P. En effet, puisque la fonction que l'on veut approximer est convexe, on a:

Proposition: L'erreur maximale sur le segment [ILIM,P] est supérieure (inférieure) à l'erreur maximale sur les n-1 segments entre P et SLIM si et seulement si $P > P^*(P^* < P)$

Cette méthode, qui est récursive, n'est pas très efficace; par exemple, pour obtenir l'approximation linéaire optimale en six morceaux de la fonction

$$y = 1000\ e^{-.0193x} \tag{1.5}$$

entre 0 et 150, il a fallu utiliser environ 40 secondes d'ordinateur (l'algorithme a été programmé en Pascal et exécuté sur l'ordinateur CDC Cyber 173 de l'Université de Montréal).

La procédure suivante trouve la meilleure approximation linéaire en n morceaux d'une fonction convexe ou concave entre ILIM et SLIM $\pm\ \delta$, où δ est la marge d'erreur tolérée sur la valeur de la borne supérieure du segment définissant le domaine sur lequel on linéarise (les paramètres d'entrée sont n, ILIM, SLIM et δ):

Algorithme LIN1

Etape 1: Choisir une valeur initiale ε pour la précision de l'approximation linéaire.

Etape 2: Trouver l'approximation linéaire en n morceaux précise à ε - près de la façon suivante:
Soit x_i la borne supérieure du i-1$^{\text{ième}}$ segment. Le point x_{i+1} est déterminé à l'aide d'une recherche dichotomique de façon à ce que l'erreur maximale sur le i$^{\text{ième}}$ segment soit ε.

Etape 3: Si $SLIM - \delta \leq x_{n+1} \leq SLIM + \delta$, terminer.
Sinon:
si $x_{n+1} < SLIM$, augmenter ε,
sinon, diminuer ε (recherche dichotomique);
aller en 2.

L'algorithme précédent est en fait une version non-récursive de l'algorithme de Geoffrion, que l'on a élaborée en relaxant la contrainte sur la borne supérieure du segment sur lequel on doit linéariser la fonction; cette contrainte est toutefois satisfaite à δ - près (il suffit donc, en pratique, de prendre δ suffisamment petit pour satisfaire la contrainte).

Pour trouver l'approximation linéaire en six morceaux de la fonction (1.5), l'algorithme LIN1 exige moins de 2 secondes (l'algorithme a été programmé en Fortran et exécuté sur l'ordinateur CDC Cyber 173 de l'Université de Montréal).

Nous présentons maintenant le problème résultant de l'approximation linéaire des fonctions de demande.

Notons par a_{ki} , $i=1,\ldots,m_k$, les pentes des m_k segments de droite formant l'approximation linéaire L_k^{-1} de U_k , et par b_{ki}, $i=1,\ldots,m_k$, les ordonnées à l'origine correspondantes; les bornes des segments seront notées x_{ki}, $i=1,\ldots,m_{k+1}$.

Soit Z_{ki} un point dans le $i^{ième}$ segment:

$$x_{ki} \leq Z_{ki} \leq x_{ki+1}$$

Alors:

$$IL_k(Z_{ki}) = \sum_{j=1}^{i-1}\left[\frac{a_{kj}x_{kj+1}^2}{2} + b_{kj}\,x_{kj+1} - \frac{a_{kj}\,x_{kj}^2}{2} - b_{kj}\,x_{kj}\right]$$

$$+ \frac{a_{ki}\,Z_{ki}^2}{2} + b_{ki}\,Z_{ki} - \frac{a_{ki}\,x_{ki}^2}{2} - b_{ki}\,x_{ki}\,,\quad 2 \leq i \leq m_{k+1} \qquad (1.6)$$

La fonction IL_k est concave puisque L_k^{-1} est décroissante. Le problème résultant, qui est convexe, est donc:

$$\text{Min } F = \sum_{ij\in A} \int_0^{f_{ij}} C_{ij}(x)dx - \sum_{k=1}^{n_p} IL_k(g_k) \qquad (P2)$$

sujet à:

$$\sum_{\ell\in\Phi_k} h_\ell = g_k \qquad (k=1,\ldots,n_p)$$

$$f_{ij} = \sum_{\ell\in\Phi} a_{ij}^{\ell}\, h_\ell \qquad (ij\in A)$$

$$h_\ell \geq 0 \qquad (\ell\in\Phi)$$

où IL_k est évalué à l'aide de (1.6).

Ce problème peut être solutionné par la méthode de Frank et Wolfe (voir [14, algorithme B2]), ce qui évite l'énumération des chemins.

Nous énonçons maintenant un théorème qui nous permet de calculer une borne supérieure sur l'erreur commise lorsque l'on résoud le problème (P2) plutôt que (P1).

Théorème. Soit L_k^{-1} une approximation linéaire de l'inverse de la fonction de demande D_k précise à $\tilde{\varepsilon}$ - près:

$$|\mu - L_k^{-1}(D_k(\mu))| \leq \tilde{\varepsilon},\ \mu \in [ILIM, SLIM],\ k=1,\ldots,n_p.$$

Soit (g*, h*) une solution ε-optimale de (P2) obtenue en utilisant L_k^{-1}. Alors (g*, h*) est une solution $(\varepsilon+\tilde{\varepsilon})$-optimale de (P1).
(Note: une solution est dite ε-optimale lorsqu'elle satisfait les conditions de Kuhn-Tucker à ε-près.)

Démonstration:

On a:

$$L_k^{-1}(g_k^*) - \Pi_k \leq \varepsilon \qquad (k=1,\ldots,n_p)$$

d'où:

$$L_k^{-1}(g_k^*) \leq \varepsilon + \Pi_k.$$

D'autre part:

$$|U_k(g_k^*) - L_k^{-1}(g_k^*)| \leq \tilde{\varepsilon}$$

d'où:

$$U_k(g_k^*) - \tilde{\varepsilon} \leq L_k^{-1}(g_k^*) \leq \varepsilon + \Pi_k$$

et finalement:

$$U_k(g_k^*) - \Pi_k \leq \varepsilon + \tilde{\varepsilon}$$

ce qui démontre que la relation (1.3) est vérifiée à $\varepsilon+\tilde{\varepsilon}$ - près. Les trois autres conditions de Kuhn-Tucker se vérifient de la même façon et

nous en omettons les démonstrations. C.Q.F.D.

Le théorème précédent donne donc une borne supérieure sur l'erreur induite sur l'équilibre du trafic par l'approximation linéaire des fonctions de demande. Il faut remarquer que si la précision de l'approximation linéaire n'est pas la même pour chacune des fonctions, i.e. si L_k^{-1} est précise à $\tilde{\varepsilon}_k$ près, alors le théorème précédent demeure valide en prenant

$$\tilde{\varepsilon} = \max_k \{\tilde{\varepsilon}_k\} \quad .$$

1.2.2.2. Modèle mathématique obtenu en utilisant les fonctions de demande

Même lorsque les inverses des fonctions de demande ne sont pas connues, il est possible d'obtenir une formulation mathématique du problème de la recherche d'un flot d'équilibre. Pour cela, il suffit de remarquer que:

$$\int_0^{g_k} U_k(x)dx = \int_{\mu_k}^{\infty} L_k(x)dx + \mu_k D_k(\mu_k)$$

où $\mu_k = U_k(g_k)$ (voir figure 1.1).

Or:

$$\int_{\mu_k}^{\infty} D_k(x)dx = \int_0^{\infty} D_k(x)dx - \int_0^{\mu_k} D_k(x)dx = \text{constante} - \int_0^{\mu_k} D_k(x)dx$$

Ainsi, le problème (P1) peut s'écrire:

$$\text{Min } F = \sum_{ij\in A} \int_0^{f_{ij}} C_{ij}(x)dx + \sum_{k=1}^{n_p} [\int_0^{\mu_k} D_k(x)dx - \mu_k D_k(\mu_k)] \qquad (P3)$$

sujet à

$$\sum_{\ell\in\Phi_k} h_\ell = D_k(\mu_k) \qquad (k=1,\ldots,n_p)$$

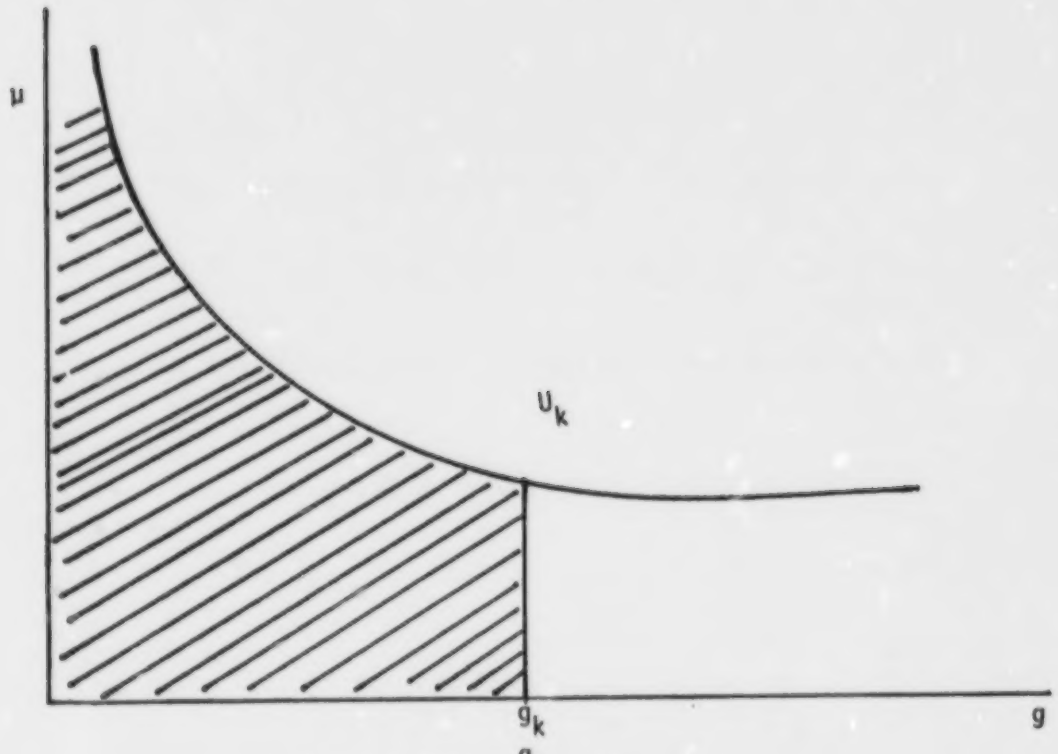


Figure 1.1-a: $\int_0^{g_k} U_k(x)dx$

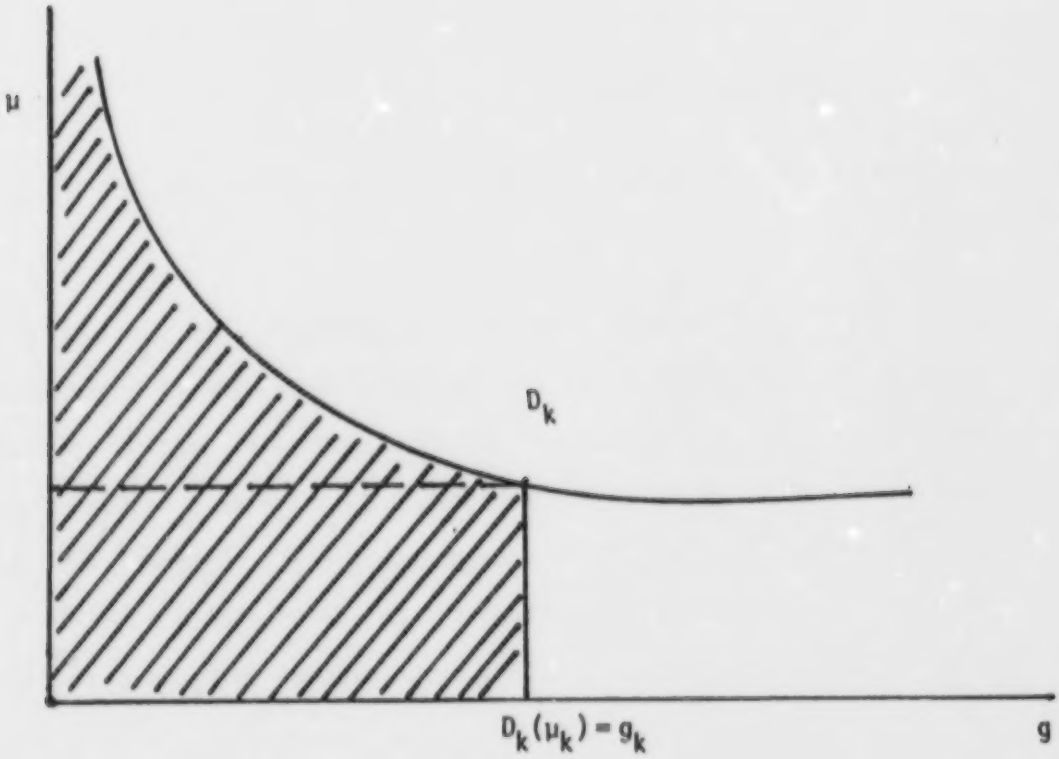


Figure 1.1-b: $\int_{\mu_k}^{\infty} D_k(x)dx + \mu_k D_k(\mu_k)$

$$\mu_k \geq 0 \qquad (k=1,\ldots,n_p)$$

$$h_\ell \geq 0 \qquad (\ell\in\Phi_k,\ k=1,\ldots n_p)$$

$$f_{ij} = \sum_{\ell\in\Phi} a_{ij}^{\ell} h_\ell \qquad (i,j)\in A$$

Il faut cependant remarquer que le problème (P3) n'est pas convexe. La fonction

$$\chi_k(\mu_k) = \int_0^{\mu_k} D_k(x)dx - \mu_k\, D_k(\mu_k)$$

est néanmoins *pseudo-convexe* croissante.

En effet, on a:

$$\chi_k'(x) = -x\, D_k'(x) \geq 0 \quad (\text{car } D_k \text{ est décroissante});$$

donc:

$$\chi_k'(y)\,(x-y) \geq 0 \Rightarrow x \geq y \Rightarrow \chi_k(x) \geq \chi_k(y) \quad \text{pour tout } x,y \geq 0$$

donc χ_k est pseudo-convexe.

Il convient aussi de noter que les contraintes de (P3) ne sont pas linéaires.

Un minimum local x^0 de (P3) satisfait les conditions de Kuhn-Tucker si et seulement si les gradients des contraintes saturées sont linéairement indépendants [11,12]. On dit alors que x^0 est un point stationnaire [12]. Dénotons par C_k l'ensemble des gradients des contraintes se rapportant à la $k^{\text{ième}}$ paire O/D et par $\overline{C}_k$ le sous-ensemble de C_k contenant seulement les gradients des contraintes saturées. Puisqu'un chemin h_ℓ donné ne relie qu'une et une seule paire O/D, aucun élément de $\overline{C}_k$ ne peut s'ex-

primer comme une combinaison linéaire d'éléments de $\bar{C}_i$ (pour k, i=1,...,n_p et k≠i); il nous suffit donc de montrer que les éléments de $\bar{C}_k$ (pour k=1,...,n_p) sont linéairement indépendants entre eux. Or, les éléments d'un ensemble $\bar{C}_k$ sont linéairement dépendants entre eux si et seulement si les gradients de toutes les contraintes se rapportant à la $k^{ième}$ paire O/D sont présents (ceci se constate aisément en vérifiant que l'ensemble engendré par les gradients de toutes les contraintes de la $k^{ième}$ paire O/D est de dimension $|\Phi_k|+2$); mais ceci ne peut se produire car si $h_\ell = 0$ pour tout $\ell \in \Phi_k$, alors $D_k(\mu_k) = 0$, donc $\mu_k > 0$. Tout minimum local de (P3) est donc un point stationnaire.

Soit (μ^o, h^o) un point stationnaire de (P3) où $\mu^o = (\mu_1^o, \mu_2^o, \ldots, \mu_{n_p}^o)$ et où h^o est le vecteur des flots sur les chemins utilisés. Par les considérations précédentes, les conditions nécessaires de Kuhn-Tucker pour le problème (P3) sont:

il existe $\Pi_k \geq 0$, k=1,...,n_p, tels que:

$$\sum_{ij \in A} a_{ij}^{\ell} c_{ij} - \Pi_k \geq 0 \qquad (\ell \in \Phi_k) \tag{1.7}$$

$$\sum_{ij \in A} a_{ij}^{\ell} c_{ij} - \Pi_k = 0 \qquad \text{si } h_\ell^o > 0 \ (\ell \in \Phi_k) \tag{1.8}$$

$$D_k'(\mu_k^o)(\mu_k^o - \Pi_k) \geq 0 \tag{1.9}$$

$$D_k'(\mu_k^o)(\mu_k^o - \Pi_k) = 0 \qquad \text{si } \mu_k^o > 0 \tag{1.10}$$

Puisque D_k est strictement décroissante, les relations (1.9) et (1.10) deviennent:

$$\mu_k^o - \Pi_k \leq 0 \qquad (1.11)$$

$$\mu_k^o - \Pi_k = 0 \qquad \text{si } \mu_k^o > 0 \qquad (1.12)$$

En posant $g_k = D_k(\mu_k^o)$ pour tout $k=1,\ldots,n_p$, les conditions (1.1) - (1.4) pour le problème (P1) sont évidemment équivalentes aux conditions (1.7), (1.8), (1.11) et (1.12) pour le problème (P3). De plus, (P1) étant convexe, tout minimum local de (P3) est une solution optimale de (P1). Ainsi, il suffit de trouver une solution quelconque de (P3) satisfaisant (1.7), (1.8), (1.11) et (1.12), et cette solution sera le flot d'équilibre cherché.

Aux quatrième et cinquième chapitres nous proposons plusieurs algorithmes de résolution du problème (P3) basés sur les méthodes des directions réalisables, du gradient réduit généralisé et de Benders.

DEUXIÈME CHAPITRE

MÉTHODES GÉNÉRALES DE RÉSOLUTION DU PROBLÈME D'AFFECTATION DU TRAFIC

2.1. Méthode des directions réalisables

De nombreux algorithmes solutionnant le problème d'affectation du trafic avec demande fixe ou variable sont basés sur la méthode des directions réalisables de Zoutendijk [14,18].

On utilisera la méthode du gradient réduit généralisé, qui est un cas particulier de la méthode de Zoutendijk, pour résoudre le problème (P3) (quatrième chapitre). Au troisième chapitre, on résoud le problème (P1) en appliquant l'algorithme des directions réalisables à un problème de demande fixe. Enfin, au cinquième chapitre, la méthode de Benders fait appel à ce même algorithme pour résoudre, là aussi, une suite de problèmes avec demande fixe. Il convient donc d'exposer dès maintenant le principe de la méthode des directions réalisables. L'exposé qui suit s'inspire fortement des présentations faites dans les références [14,18].

Soit X^0 une solution initiale (réalisable) à un programme non-linéaire donné. La méthode des directions réalisables consiste à calculer une suite $\{X^i\}$ de solutions réalisables telles que chaque nouvelle solution améliore la valeur de la fonction objectif. Chacun des X^i dans la suite est obtenu à partir de X^{i-1} en déterminant une direction d'amélioration de la fonction objectif et le déplacement à faire dans cette direction.

Soit V le domaine réalisable d'un programme non-linéaire. On dit qu'une

direction d est réalisable en $X^0 \in V$ s'il existe un $\sigma > 0$ tel que

$$X^0 + \tau d \in V \qquad \text{pour} \quad 0 \leq \tau \leq \sigma .$$

La direction d améliore la fonction objectif F à minimiser s'il existe un $\gamma > 0$ tel que

$$\nabla(X^0)'d < 0$$

c'est-à-dire

$$F(X^0 + \tau d) < F(X^0) \qquad \text{pour tout } 0 < \tau \leq \gamma$$

L'algorithme se termine lorsque la solution courante X^i satisfait un certain critère d'optimalité dépendant du problème traité.

Deux méthodes permettant de déterminer des directions de descente pour le problème (P1) (et pour sa version en demande fixe) sont données dans [14] : la méthode de Frank et Wolfe et celle du gradient réduit. C'est cette dernière qui est utilisée pour résoudre les sous-problèmes avec demande fixe dans les algorithmes L1 (troisième chapitre) et B1 (cinquième chapitre).

2.2. Décomposition par paire O/D

Les algorithmes présentés dans cette section et la suivante ont été élaborés par Sang Nguyen dans le cadre de sa thèse de doctorat [14].

Considérons le problème (P3):

$$\text{Min } F = \sum_{ij \in A} \int_0^{f_{ij}} C_{ij}(x)dx + \sum_{k=1}^{n_p} \left[\int_0^{\mu_k} D_k(x)dx - \mu_k \, D_k(\mu_k) \right]$$

sujet à

$$f_{ij} = \sum_{\ell \in \Phi} a_{ij}^{\ell} h_{\ell}$$

$$\sum_{\ell \in \Phi_k} h_{\ell} = D_k(\mu_k) \qquad (k=1,\ldots,n_p)$$

$$h_{\ell} \geq 0 \qquad (\ell \in \Phi_k,\ k=1,\ldots,n_p)$$

$$\mu_k \geq 0 \qquad (k=1,\ldots n_p)$$

On note que les variables décrivant des paires origine-destination différentes ne sont reliées entre elles que par la fonction objectif. Nguyen [14] a démontré qu'on peut décomposer le problème (P3) en n_p sous-problèmes $SP_1,\ldots SP_{n_p}$, chacun des sous-problèmes ne comportant qu'une paire O/D, et résoudre (P3) en résolvant les n_p sous-problèmes un à un, de façon cyclique.

L'avantage d'utiliser une telle méthode est que chacun des sous-problèmes est moins gros que le problème originel et, ne comportant qu'une seule contrainte, plus facile à résoudre. De plus, on peut ne conserver en mémoire centrale que les informations relatives à la paire O/D que l'on traite.

Cependant, l'inconvénient majeur de cette méthode par rapport à une méthode de type Frank et Wolfe, permettant une approche globale, est la grande quantité d'informations qu'il faut conserver: en effet, toutes les méthodes basées sur la décomposition par paire O/D exigent que l'on connaisse la description de chacun des chemins utilisés en termes d'arcs. De plus, dû au fait que l'on résoud le problème originel en résolvant un certain nombre de fois, et de façon cyclique, les n_p sous-problèmes, on ne connaît pas le taux de convergence exact de cette classe d'algorithmes.

La procédure permettant la décomposition par paire origine-destination est la suivante:

Algorithme D1

Etape 1: Trouver une affectation du trafic quelconque pour chacune des paires O/D.

Poser ITER ← 0 et EQUIL ← 0 .

Etape 2: Si EQUIL = n_p terminer; sinon, poser:

$k \leftarrow ITER \bmod (n_p) + 1$

Etape 3: Si la solution courante est un trafic d'équilibre pour la $k^{ième}$ paire O/D, poser:

EQUIL ← EQUIL + 1

ITER ← ITER + 1

et aller en 2; sinon, trouver le flot d'équilibre pour la $k^{ième}$ paire O/D, faire

EQUIL ← 1

ITER ← ITER + 1

et aller en 2.

2.3. Génération progressive des chemins utilisés

La formulation du problème (P3) suppose que l'on connaît Φ_k, l'ensemble de tous les chemins reliant la $k^{ième}$ paire O/D (pour $k=1,\ldots,n_p$). Or identifier, codifier et conserver en mémoire tous ces chemins peut s'avérer très coûteux; heureusement, il n'est pas nécessaire de les connaître tous.

Soit Φ_k^0 un sous-ensemble de Φ_k comportant au moins un chemin. Supposons que nous ayons un flot et un coût d'équilibre $\{h_\ell^*, \mu_k^*\}$ sur les chemins de Φ_k^0. Alors les conditions de Kuhn-Tucker sont vérifiées, i.e.

Il existe $\Pi_k \geq 0$ tel que

$$\alpha_\ell \geq \Pi_k \qquad \text{(pour tout } \ell \in \Phi_k^0\text{)}$$

$$\alpha_\ell = \Pi_k \qquad \text{si } h_\ell^* > 0$$

$$\mu_k^* \geq \Pi_k$$

$$\mu_k^* = \pi_k \qquad \text{si } \mu_k^* > 0,$$

où

$$\alpha_\ell = \sum_{ij \in A} a_{ij}^\ell \, C_{ij}(f_{ij}) \ ,$$

$[a_{ij}^\ell]$ étant la matrice d'incidence arc-chemin.

Comme on a $h_\ell^* = 0$ lorsque $\ell \notin \Phi_k^0$, si

$$\alpha_\ell \geq \mu_k \qquad \text{pour tout } \ell \in \Phi_k - \Phi_k^0 \tag{2.1}$$

alors le flot et le coût $\{h_\ell^*, \mu_k^*\}$ réalisent un trafic d'équilibre sur l'ensemble Φ_k de tous les chemins possibles.

Pour vérifier (2.1) il suffit, bien sûr, de vérifier que le coût de parcours μ_{min} du chemin à coût minimum reliant la $k^{ième}$ paire O/D est supérieur ou égal à μ_k. D'autre part, si

$$\mu_{min} < \mu_k$$

on ajoute à Φ_k^0 le chemin correspondant et on cherche un nouvel équilibre.

On peut donc résoudre (P3) par la méthode de décomposition en générant progressivement les chemins à coût de parcours minimum. Plus formellement:

Algorithme G1

Etape 1: Pour chaque paire O/D, on a un ensemble de chemins initial Φ_k et un coût μ_k. Trouver une affectation de trafic quelconque h_ℓ, $\ell \in \Phi_k^0$, telle que

$$\sum_{\ell \in \Phi_k^0} h_\ell = D_k(\mu_k) .$$

Poser $m \leftarrow 0$.

Etape 2: Résoudre le problème d'affectation sur l'ensemble des chemins disponibles

$$\Phi^m = \bigcup_k \Phi_k^m$$

Soient μ_k, $k=1,\ldots,n_p$ les coûts à l'équilibre.

Etape 3: Pour chaque paire O/D d'indice k, trouver un chemin b_k à coût minimum μ_{b_k} et faire

$$\Phi_k^{m+1} \leftarrow \begin{cases} \Phi_k^m \cup \{b_k\} & \text{si } \mu_{b_k} > \mu_k \\ \Phi_k^m & \text{sinon} \end{cases}$$

Si $\Phi^{m+1} = \bigcup_k \Phi_k^{m+1} \subseteq \Phi^m$ terminer; sinon, faire $m \leftarrow m+1$ et aller en 2.

TROISIÈME CHAPITRE

UNE MÉTHODE DE RÉSOLUTION BASÉE SUR L'APPROXIMATION LINÉAIRE PROGRESSIVE DES FONCTIONS DE DEMANDE

3.1. Un algorithme d'approximation linéaire

La méthode décrite dans ce chapitre résoud directement le problème (P1):

$$\text{Min } F = \sum_{ij\in A} \int_0^{f_{ij}} C_{ij}(x)dx \; - \sum_{k=1}^{n_p} \int_0^{g_k} U_k(x)dx$$

$$\text{s.a.} \quad \sum_{\ell\in\Phi_k} h_\ell = g_k \qquad (k=1,\ldots,n_p)$$

$$h_\ell \geq 0 \qquad (j\in\Phi)$$

$$f_{ij} = \sum_{\ell\in\Phi} a_{ij}^{\ell}\, h_\ell \qquad (ij\in A)$$

même lorsque les fonctions U_k ne sont pas connues; les fonctions D_k doivent être continues et strictement décroissantes.

La méthode habituellement utilisée pour résoudre ce problème lorsque les U_k ne sont pas connues est la suivante:

Etape 1: Prendre un point initial (μ^0, g^0).

Etape 2: Faire une affectation avec demande fixe g^0 pour obtenir (μ', g^0).
Si $|\mu^0-\mu'| < \varepsilon$ terminer. ($\varepsilon > 0$ est une tolérance pré-déterminée).

Etape 3: Calculer $g' \leftarrow D_k(\mu')$

Etape 4: Faire une affectation avec demande fixe g' pour obtenir μ^0.
Faire $g^0 \leftarrow g'$, aller en 2.

Cette méthode, qui s'applique sur une paire O/D et doit donc être utilisée avec la méthode de décomposition (section 2.2), ne converge pas toujours. Pour s'en convaincre, il suffit de considérer la figure 3.1, montrant une itération typique de l'algorithme.

Nous présentons ici une méthode convergente. L'idée de base de cette méthode est d'approximer la fonction de demande par un segment de droite $[(\mu^i, g^i), (\mu^{i+1}, g^{i+1})]$ (où $g^i = D_k(\mu^i)$) tel que le point d'équilibre (μ^*, g^*) soit contenu entre les points extrêmes de ce segments, i.e.

$$\mu^i \leq \mu^* \leq \mu^{i+1} \quad \text{et} \quad g^{i+1} \leq g^* \leq g^i.$$

On peut alors approximer U_k par ce segment et résoudre un problème avec demande variable. On démontrera que si la fonction de demande est décroissante, la distance entre la projection sur un axe donné de chacun des deux points bornant le segment diminue à chaque étape.

La procédure pour trouver le point d'équilibre de la $k^{ième}$ paire O/D est la suivante:

Algorithme L1

Etape 1: Prendre un coût initial arbitraire μ_0^0, calculer $g_0^0 \leftarrow D_k(\mu_0^0)$.
Poser $i \leftarrow 0$.

Etape 2: Affecter la demande g_0^i sur le réseau; ceci donnera le coût de transport à l'équilibre μ_1^i .

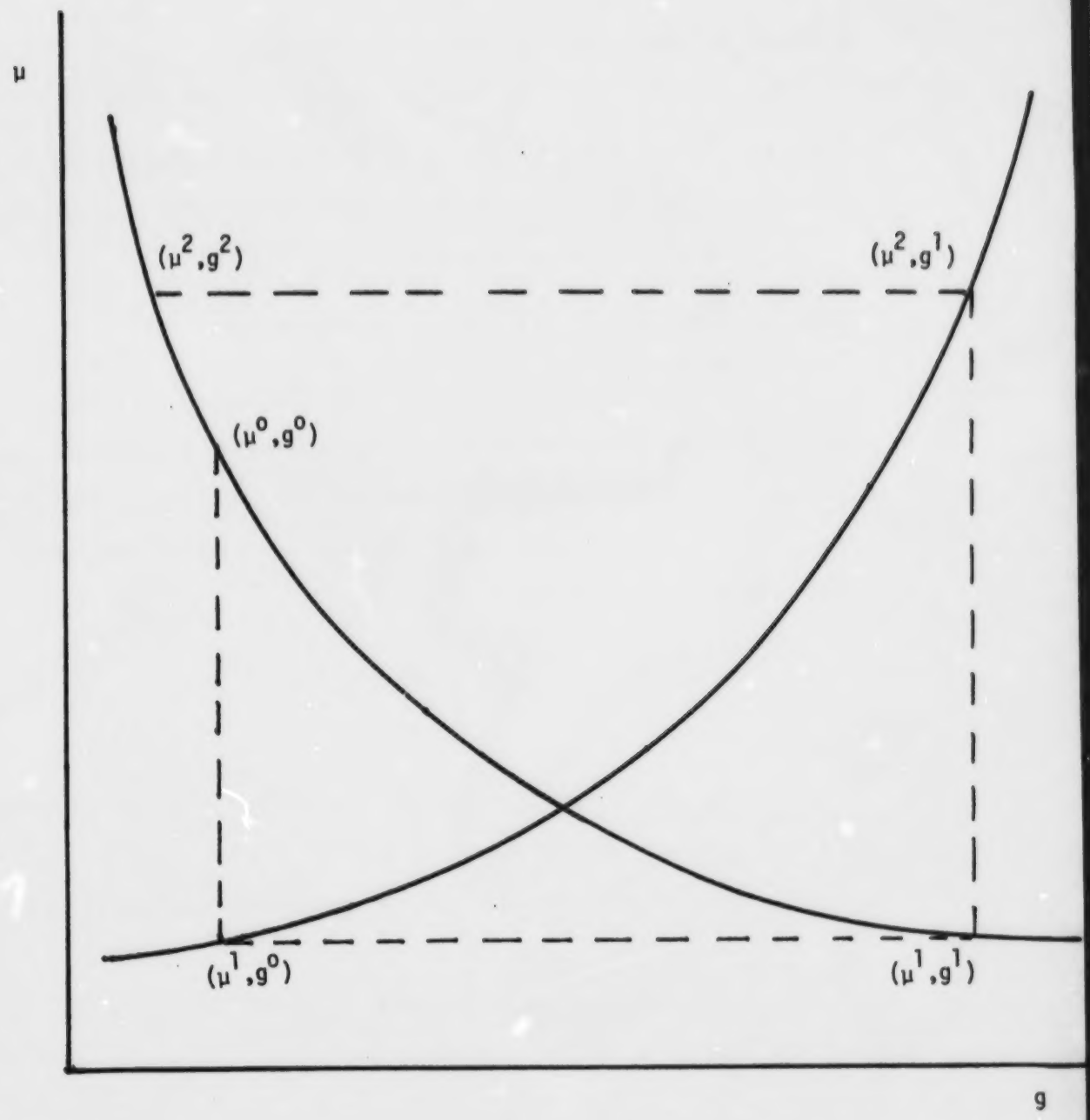


Figure 3.1: Méthode généralement utilisée pour résoudre (P1) lorsque les inverses ne sont pas connues

Etape 3: Calculer $g_1^i \leftarrow D_k(\mu_1^i)$.

Si $|g_1^i - g_0^i| \leq \varepsilon$ terminer ($\varepsilon > 0$ est une tolérance pré-déterminée).

Etape 4: L'inverse de la fonction de demande est approximée par la droite $L[(\mu_0^i, g_0^i), (\mu_1^i, g_1^i)]$; résoudre le problème de demande variable et obtenir le point d'équilibre $(\bar{\mu}^i, \bar{g}^i)$.

Etape 5: Poser $i \leftarrow i+1$, $\mu_0^i \leftarrow \bar{\mu}^{i-1}$, $g_0^i \leftarrow D_k(\mu_0^i)$

Aller en 2.

Aux étapes 2 et 4, les points d'équilibre peuvent être déterminés par l'étape 3 des algorithmes A1 et B1 de [14] respectivement. Lorsque les coûts $C_{ij}(x)$ sur les arcs sont linéaires, le calcul du pas optimal dans l'algorithme B1 [14] se réduit à:

$$- \sum_{ij \in A} c_{ij} d_{ij} + pgd_g \Big/ \sum_{ij \in A} a_{ij} d_{ij}^2 - pgd_g^2$$

où

$c_{ij} = a_{ij} f_{ij} + b_{ij}$ est le coût sur l'arc ij;

d_{ij} est la direction sur l'arc ij;

p est la pente de la droite qui approxime l'inverse de la fonction de demande;

g est la demande de déplacement interzonal et

d_g est la direction associée à g.

Un algorithme itératif qui, à partir d'une solution réalisable $z' \in V$ génère une suite de solutions réalisables $\{z^k (k=1,\ldots,\infty,\ z^k \in V)\}$ peut être représenté par une application $A: V \rightarrow V$ qui à tout point z^k de la suite associe un ensemble $A(z^k)$ tel que

$$z^{k+1} \in A(z^k)$$

(voir [17]).

Le théorème suivant, dû à Zangwill [17], établit les conditions nécessaires qui assurent la convergence en programmation non-linéaire.

Théorème: Soit une application $A : V \to V$ représentant un algorithme itératif qui, à partir d'une solution initiale $z' \in V$ génère la suite de solutions réalisables $\{z^k, k=1,\ldots,\infty\}$ où $z^{k+1} \in A(z^k)$.

Supposons que:

a) toutes les solutions z^k sont dans un compact $X \subset V$.

b) il existe une fonction continue $G : V \to R$ telle que

i) si z n'est pas une solution optimale, alors pour tout $y \in A(z)$ on a $G(y) < G(z)$

ii) si z est une solution optimale, alors ou bien l'algorithme termine (i.e. A(z) est vide) ou bien pour tout $y \in A(z)$ on a $G(y) \geq G(z)$.

c) l'application A est fermée en tout point qui n'est pas une solution optimale.

Alors, ou bien l'algorithme termine à une solution optimale, ou bien toute valeur d'adhérence de la suite $\{z^k\}$ est une solution optimale

La démonstration de ce théorème se trouve dans Zangwill [17].

Avant de démontrer que l'algorithme L1 converge, il faut rappeler que:

a) une fonction continue est une application fermée [17] et que

b) la composition de deux applications fermées est une application fermée [14,17].

Lemme. Pour tout i, $|g_1^{i+1} - g_0^{i+1}| < |g_1^i - g_0^i|$

Démonstration

A l'itération i, en supposant que l'équilibre n'est pas atteint, on peut avoir:

$$\mu_0^i > \mu^* \quad \text{ou} \quad \mu_0^i < \mu^*.$$

Traitons le cas où $\mu_0^i > \mu^*$. (La preuve est tout à fait similaire si l'on considère l'autre cas)

Soit $S(g)$ la fonction (croissante) donnant le coût à l'équilibre lorsqu'on affecte une demande de g unités sur le réseau.

On a:

$$\mu_0^i > \mu^* \Rightarrow g_0^i < g^* \qquad \text{car } D_k \text{ est décroissante}$$

et $$g_0^i < g^* \Rightarrow \mu_1^i < \mu^* \qquad \text{car } S \text{ est croissante}$$

où $\mu_1^i = S(g_0^i)$ est le coût d'équilibre lorsque la demande est fixée à g_0^i.

Soit $g_1^i = D_k(\mu_1^i)$. Alors:

$$\mu_1^i < \mu^* \Rightarrow g_1^i > g^*.$$

Pour résumer, on a donc:

$$\mu_1^i < \mu^* < \mu_0^i \tag{3.1}$$

et $$g_0^i < g^* < g_1^i \tag{3.2}$$

Notons par $\bar{D}^{-1}$ l'approximation linéaire de l'inverse de la fonction de demande. On note par $(\bar{\mu}^i, \bar{g}^i)$ le point d'équilibre (demande variable) obtenu avec l'inverse $\bar{D}^{-1}$. On définit:

$$\mu_0^{i+1} = \bar{\mu}^i$$

et $$g_0^{i+1} = D_k(\mu_0^{i+1})$$

On a alors une des deux situations suivantes:

$$\mu_0^{i+1} > \mu^* \quad \text{ou} \quad \mu_0^{i+1} < \mu^*$$

(l'égalité est exclue puisque $(\bar{\mu}^i, \bar{g}^i)$ est obtenue avec $\bar{D}^{-1}$). Traitons d'abord le premier cas.

La fonction D_k étant décroissante, on a:

$$g_0^{i+1} < g^* \tag{3.3}$$

Or,

$$\mu_0^{i+1} = \bar{\mu}^i < \mu_0^i \tag{3.4}$$

car $\bar{\mu}^i$ est nécessairement sur le segment $L[(\mu_0^i, g_0^i), (\mu_1^i, g_1^i)]$, l'égalité étant exclue parce que la fonction $S(g)$ ne passe par pas (μ_0^i, g_0^i) (i.e. $\mu_0^i \neq \mu^*$).

De (3.4) on tire:

$$g_0^{i+1} > g_0^i \tag{3.5}$$

D'autre part, on obtient:

$$\mu_i^{i+1} = S(g_0^{i+1})$$

en affectant la demande g_0^{i+1} et on pose:

$$g_1^{i+1} = D_k\ (\mu_1^{i+1}).$$

De (3.5) et du fait que S est croissante on tire:

$$\mu_1^{i+1} > \mu_1^i \tag{3.6}$$

Donc, puisque D_k est décroissante,

$$g_1^{i+1} < g_1^i \tag{3.7}$$

De plus, comme $\mu_0^{i+1} > \mu^*$ on a

$$\mu_1^{i+1} < \mu^* \tag{3.8}$$

et $$g_1^{i+1} > g^* \tag{3.9}$$

Combinant (3.2), (3.3), (3.5), (3.7) et (3.9), on tire:

$$|g_1^{i+1} - g_0^{i+1}| < |g_1^i - g_0^i|$$

ou, combinant (3.1), (3.4), (3.6) et (3.8):

$$\mu_1^{i+1} - \mu_0^{i+1} < \mu_1^i - \mu_0^i$$

En utilisant les mêmes arguments, on prouve facilement, dans le cas où

$$\mu_0^{i+1} < \mu^*,$$

les énoncés suivants:

$$\mu_0^{i+1} < \mu^* < \mu_1^{i+1}$$
$$\mu_0^{i+1} > \mu_1^i$$
$$\mu_1^{i+1} < \mu_0^i$$

d'où l'on tire:

$$g_1^{i+1} < g^* < g_0^{i+1}$$
$$g_1^{i+1} > g_0^i$$
$$g_0^{i+1} < g_1^i$$

et finalement

$$|g_1^{i+1} - g_0^{i+1}| < |g_1^i - g_0^i| \qquad \text{C.Q.F.D.}$$

Théorème: L'algorithme L1 est convergent.

Démonstration:

Il suffit de démontrer que les trois conditions spécifiées dans le théorème de convergence de Zangwill sont respectées.

a) Par le lemme précédent, l'ensemble des solutions réalisables généré est inclus dans $[\mu_1^0,\mu_0^0] \times [g_0^0,g_1^0]$ qui est un ensemble compact.

b) Définissons

$$G(g_0^i,g_1^i) = |g_1^i - g_0^i| \ .$$

Cette fonction est évidemment continue.

i) par le lemme précédent, la fonction G diminue strictement à chacune des itérations;

ii) l'algorithme termine lorsque la solution optimale est atteinte.

c) Une itération de l'algorithme L1 se divise en 4 étapes.

i) On a un point initial (μ_0^i,g_0^i). On solutionne le problème d'affectation du traffic avec demande fixe à l'aide de la méthode du gradient réduit pour obtenir (μ_1^i,g_0^i). Or, il a été démontré [14] qu'il existe une application fermée A1 correspondant à l'algorithme du gradient réduit.

ii) On obtient (μ_1^i,g_1^i) à l'aide de la fonction de demande D. Or, on a fait remarquer plus haut qu'une fonction continue est une application fermée.

iii) Le point d'équilibre $(\mu_0^{i+1},\bar{g}^i)$ est obtenu en solutionnant le

problème d'affectation du trafic avec demande variable en utilisant la méthode du gradient réduit. Il existe donc une application A2 correspondant à cette étape.

iv) On utilise la fonction de demande D pour obtenir (μ_0^{i+1}, g_0^{i+1}).

L'application A correspondant à l'algorithme L1 est donc composée de quatre applications fermées:

$$A = D\ A2\ D\ A1.$$

L'application A est donc fermée et l'algoritme L1 est convergent. C.Q.F.D.

La figure 3.2, montrant une itération typique de l'algorithme, permettra de fixer les idées.

Notons que cet algorithme est conceptuellement similaire à la méthode de la sécante pour trouver la racine d'une fonction [12].

3.2. Amélioration de la méthode

L'efficacité de l'algorithme L1 peut être améliorée en remplaçant l'étape 5 par les étapes suivantes:

Algorithme L2

Etape 5': Calculer $\bar{g}'^{i} \leftarrow D_k(\bar{\mu}^i)$

Etape 6: Soit (μ_2^i, g_2^i) l'intersection des segments:
$L[(\bar{\mu}^i, \bar{g}'^{i}), (\mu_1^i, g_1^i)]$ et $L[(\mu_1^i, g_0^i), (\bar{\mu}^i, \bar{g}^i)]$.

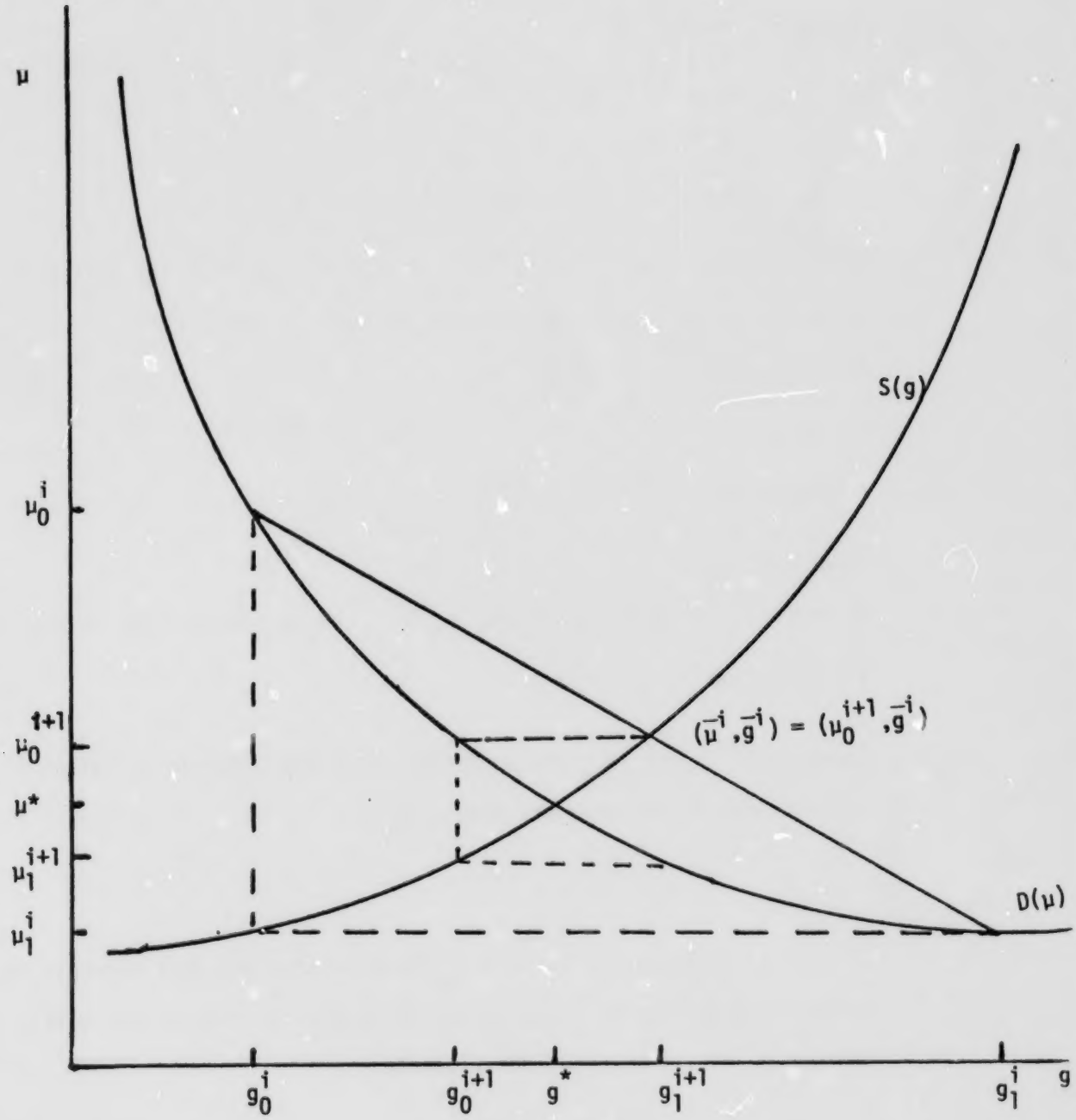


Figure 3.2: Itération typique de l'algorithme L1

Etape 7: Poser:

$$i \leftarrow i+1, \quad \mu_0^i \leftarrow \mu_2^{i-1}, \quad g_0^i \leftarrow D_k(\mu_0^i).$$

Aller en 2.

Cette procédure est illustrée par la figure 3.3.

On dénotera par $g_0^{i(L1)}$, $g_1^{i(L1)}$ et $g_0^{i(L2)}$, $g_1^{i(L2)}$ les suites de points obtenues par les algorithmes L1 et L2 respectivement.

Proposition:

$$|g_1^{i(L2)} - g_0^{i(L2)}| < |g_1^{i(L1)} - g_0^{i(L1)}|, \quad i \neq 0$$

en supposant $g_0^{0(L2)} = g_0^{0(L1)}$

Démonstration

On supposera le point de départ $\mu_0^0 > \mu^*$. On ne traitera que le cas où, à l'itération i, $\mu_0^{i(L1)} > \mu^*$.

Comme $\bar{\mu}^{i(L1)}$ est la borne supérieure d'un des segments à l'étape 6 (voir preuve du théorème précédent) on a:

$$\mu_0^{i+1(L2)} = \mu_2^{i(L2)} < \bar{\mu}^{i(L1)}$$

(l'égalité étant exclue parce que l'intersection des deux segments ne se situe pas à une borne, sinon ladite borne est la solution optimale), ou, puisque D_k est strictement décroissante:

$$g_0^{i+1(L2)} > g_0^{i+1(L1)} \tag{3.10}$$

D'autre part

$$\mu_2^{i(L2)} = \mu_0^{i+1(L2)} > \mu^* \tag{3.11}$$

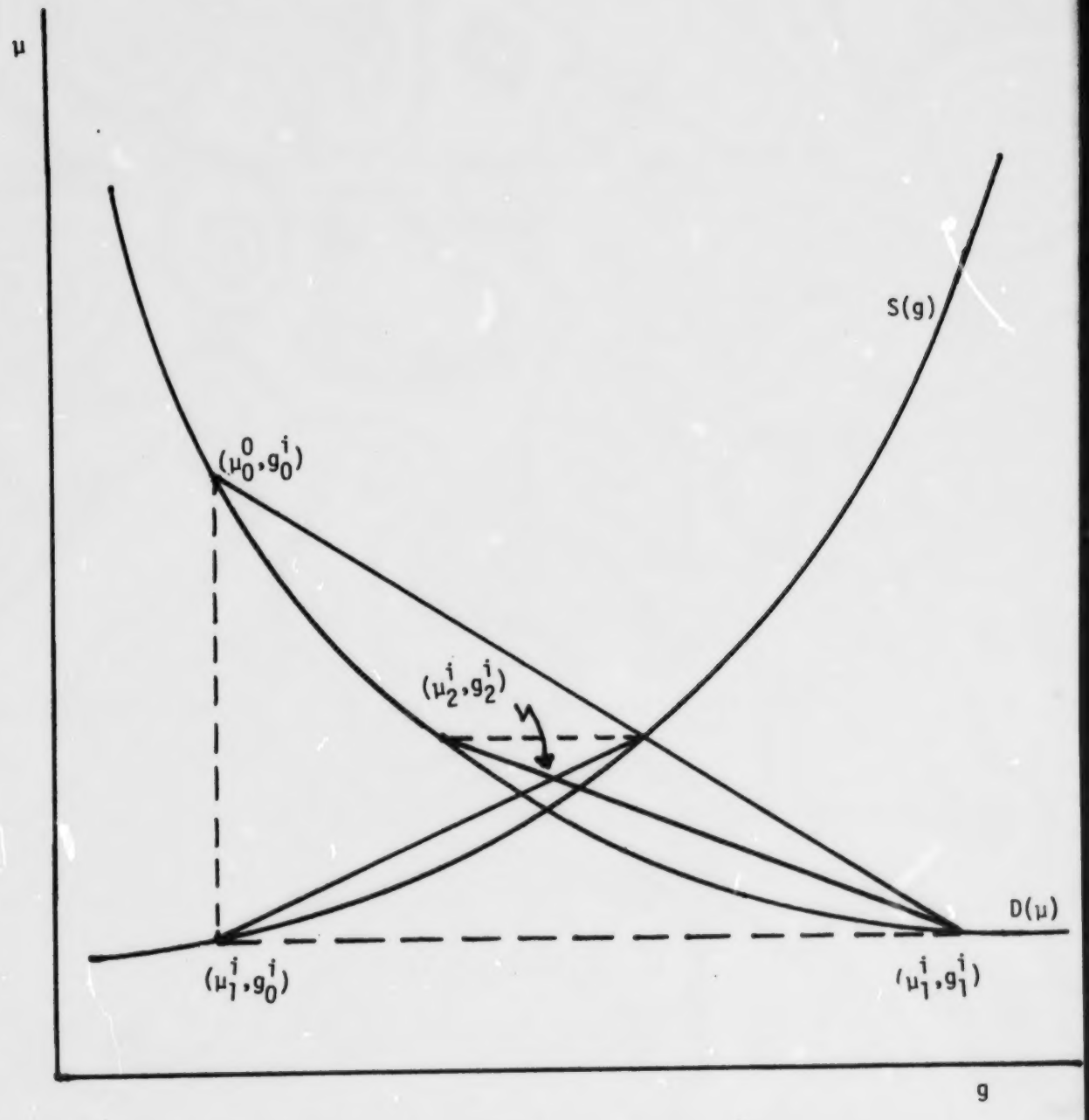


Figure 3.3: Itération typique de l'algorithme L2

d'où:

$$g_0^{i+1(L2)} < g^* \tag{3.12}$$

D'autre part, par (3.11), et puisque la fonction S(g) est croissante:

$$S(g_0^{i+1(L2)}) = \mu_1^{i+1(L2)} > \mu_1^{i+1(L1)} \tag{3.13}$$

d'où:

$$g_1^{i+1(L2)} < g_1^{i+1(L1)} \tag{3.14}$$

Puis, de (3.13)

$$\mu_1^{i+1(L2)} < \mu^* \tag{3.15}$$

par croissance de S, et:

$$g_1^{i+1(L2)} > g^* \tag{3.16}$$

Finalement, on sait, par la démonstration du théorème précédent, que,

$$\mu_1^{i+1(L1)} < \mu^* < \mu_0^{i+1(L1)} \tag{3.17}$$

et que

$$g_0^{i+1(L1)} < g^* < g_1^{i+1(L1)} \tag{3.18}$$

De (3.11), (3.13), (3.15), (3.17) et (3.19):

$$|g_1^{i+1(L2)} - g_0^{i+1(L2)}| < |g_1^{i+1(L1)} - g_0^{i+1(L1)}| \qquad \text{C.Q.F.D.}$$

On peut interpréter intuitivement la proposition précédente en disant que l'algorithme L2 est plus efficace que L1 puisqu'à chaque itération la diminution de la longueur de l'intervalle est plus importante.

QUATRIÈME CHAPITRE

ALGORITHMES BASES SUR LA MÉTHODE DES DIRECTIONS RÉALISABLES

4.1 La méthode du gradient réduit généralisé

On présente dans cette section une extension, dûe à Abadie et Carpentier [1], de la méthode du gradient réduit de Wolfe au cas de problèmes avec contraintes non-linéaires.

Soit le problème suivant:

$$\text{Min } f(X) \tag{4.1}$$

$$\text{s.a.} \quad g(X) = 0\ , \quad X \geq 0 \tag{4.2}$$

où $X \in R^N$, $g: R^N \to R^m$ et f et g sont continuement différentiables.

On dénote par V le domaine réalisable:

$$V = \{X \in R^N : g(X) \quad 0,\ X \geq 0\}\ .$$

Soit $X^0 \in R^N$. On suppose qu'il y a non-dégénérescence, c'est-à-dire qu'il existe une partition (x^0, y^0) de X^0 telle que:

$$y^0 > 0 \quad (y^0 \in R^m,\ x^0 \in R^n = R^{N-m})$$

et que la matrice

$$[\partial g/\partial y]_{X = X^0}$$

est non singulière (inversible). On peut facilement vérifier que ces conditions sont toujours satisfaites par le problème (P3).

On réécrit alors le problème (4.1) - (4.2):

$$\text{Min} \quad f(x,y) \tag{4.3}$$

$$\text{s.a.} \quad g(x,y) = 0 \tag{4.4}$$

$$x \geq 0,\ y \geq 0 \tag{4.5}$$

Les conditions de Kuhn-Tucker pour le problème (4.3) - (4.5) sont:

Il existe $\mu^* \in R^m$ et $v^t \in R^n$ tels que

$$v_i \begin{cases} \leq 0 & \text{si } x_i = 0 \\ = 0 & \text{si } x_i > 0 \end{cases} \tag{4.6}$$

$$-\frac{\partial f}{\partial x^0} + \mu \frac{\partial g}{\partial x^0} = v \tag{4.7}$$

$$-\frac{\partial f}{\partial y^0} + \mu \frac{\partial g}{\partial y^0} = 0 \tag{4.8}$$

d'où l'on tire:

$$v = -\frac{\partial f}{\partial x^0} + \frac{\partial f}{\partial y^0} \left(\frac{\partial g}{\partial y^0} \right)^{-1} \frac{\partial g}{\partial x^0} \tag{4.9}$$

(toutes ces expressions sont évaluées en X^0).

Si v, calculé selon (4.9), satisfait (4.6), alors X^0 est optimal. Si, par contre, v ne satisfait pas (4.6), on calcule une direction d selon laquelle on modifie la valeur de x^0:

$$x^1 = x^0 + \theta d \quad , \quad \theta > 0$$

et

$$d_i = \begin{cases} 0 & \text{si } x_i = 0 \text{ et } v_i < 0 \\ v_i & \text{sinon.} \end{cases}$$

La direction $k \in R^m$ associée aux variables de base y^0 sera calculée selon la tangente à la courbe

$$x = x^0 + \theta d$$

$$y = y(\theta),\ y(0) = y^0$$

$$g(x^0 + \theta d,\ y(\theta)) = 0$$

qui est

$$x = x^0 + \theta d$$

$$y = y^0 + \theta k$$

$$\frac{\partial g}{\partial x^0}\ d + \frac{\partial g}{\partial y^0}\ k = 0$$

d'où l'on tire

$$k = -\left(\frac{\partial g}{\partial y^0}\right)^{-1} \frac{\partial g}{\partial x^0}\ d \qquad (4.10)$$

Cependant, puisque la direction associée aux variables de base suit une tangente au domaine réalisable, la direction obtenue ne sera pas, en général, réalisable. Ainsi, lorsqu'on a trouvé la valeur θ^* de θ qui minimise

$$f(x^0 + \theta d,\ y^0 + \theta k)$$

tout en conservant $(x^0,\ y^0) \geq 0$, on n'aura pas, en général,

$$g(x^0 + \theta^* d,\ y^0 + \theta^* k) = 0.$$

On doit donc trouver un $\tilde{y}$ tel que

$$g(x^0 + \theta^*\ d,\ \tilde{y}) = 0$$

Cela se fait aisément par la méthode de Newton.

D'une façon plus formelle, l'algorithme est le suivant:

Algorithme GRG

Etape 1: Trouver une solution initiale X^0, poser $I \leftarrow 0$.

Etape 2: Partitionner X^I en variables de base y^I et en variables hors-base x^I.

Calculer $v \in R^n$:

$$v \leftarrow -\frac{\partial f}{\partial x^I} + \frac{\partial f}{\partial y^I}\left(\frac{\partial g}{\partial y^I}\right)^{-1}\frac{\partial g}{\partial x^I}$$

(les dérivées étant évaluées en X^I).

Poser:

$$d_i \leftarrow \begin{cases} 0 & \text{si } x_i^I = 0 \text{ et } v_i < 0 \\ v_i & \text{sinon.} \end{cases}$$

Si $d = 0$ terminer.

Calculer $k \in R^m$:

$$k \leftarrow -\left(\frac{\partial g}{\partial y^I}\right)^{-1}\frac{\partial g}{\partial x^I}\, d$$

Etape 3: Soit $\bar{\theta} \leftarrow \min_{i \mid x_i^I < 0} \{-x_i^I/d_i\}$

Trouver θ^*, $0 < \theta^* \leq \bar{\theta}$ qui minimise:

$$f(x^I + \theta d,\ y^I + \theta k).$$

Poser:

$$\bar{x} \leftarrow x^I + \theta * d$$

$$\tilde{y}^1 \leftarrow y^I + \theta * k.$$

Etape 4: Si $g(\bar{x}, \tilde{y}^1) = 0$ poser $\bar{y} \leftarrow \tilde{y}^1$, aller en 5.

Sinon, poser $j \leftarrow 1$.

4.1: Calculer:

$$\tilde{y}^{j+1} \leftarrow \tilde{y}^j - \left(\frac{\partial g}{\partial y^I}\right)^{-1} g(\bar{x}, \tilde{y}^j)$$

Si $|\tilde{y}^{j+1} - \tilde{y}^j| < \varepsilon$ ($\varepsilon > 0$ est une tolérance pré-déterminée)

poser $\bar{y} \leftarrow \tilde{y}^{j+1}$, aller en 5.

Sinon poser $j \leftarrow j+1$, aller en 4.1.

Etape 5: Poser

$$I \leftarrow I + 1$$

$$x^I \leftarrow (\bar{x}, \bar{y}).$$

Aller en 2.

(Note: l'étape 4.1 est la méthode de Newton pour trouver la racine d'une fonction.)

Cet algorithme donne une solution ε-réalisable puisque, à l'étape 4.1, on trouve une racine approximative (à ε-près) à l'équation $g(\bar{x},y)=0$. A la section 4.3, nous présentons une direction réalisable, c'est-à-dire une direction telle que la solution courante est toujours réalisable.

4.2. Adaptation de l'algorithme GRG au problème d'affectation du trafic

Le problème (P3) s'écrit, lorsqu'on ne considère que la $k^{ième}$ paire O/D:

$$\text{Min } F = \sum_{ij\in A} \int_0^{f_{ij}} C_{ij}(x)dx + \int_0^{\mu_k} D_k(x)dx - \mu_k\, D_k\,(\mu_k)$$

s.a. $$\sum_{\ell\in\Phi_k} h_\ell = D_k\,(\mu_k)$$

$$h_\ell \geq 0 \qquad (\ell\in\Phi_k)$$

$$\mu_k \geq 0$$

$$f_{ij} = \sum_{\ell\in\Phi} a_{ij}^{\ell}\, h_\ell \qquad (ij\in A)$$

C'est un problème avec une contrainte d'égalité non-linéaire (on n'a pas à tenir compte des contraintes du dernier type puisque ce ne sont que des contraintes de transformation de variables).

Les dérivées partielles sont ici très faciles à calculer. On prendra comme variable de base:

$$h_b = \max_{\ell\in\Phi_k} \{h_\ell\}\ .$$

On trouve alors:

$$v_\ell = \alpha_b - \alpha_\ell$$

$$v_\mu = D_k'\,(\mu_k)\,(\mu_k - \alpha_b)$$

où

$$\alpha_\ell = \sum_{ij\in A} a_{ij}^{\ell}\, C_{ij}\,(f_{ij})$$

est le coût du chemin ℓ. On a aussi

$$k = -\sum_{\ell \neq b} d_\ell + D_k'(\mu_k)\, d_\mu$$

Ainsi, l'algorithme GRG spécialisé pour le problème (P3) est:

Algorithme GRG1

Etape 1: Trouver une solution initiale $\{h_\ell, \ell \in \Phi_k;\ \mu_k\}$.

Poser $I \leftarrow 1$.

Etape 2: Trouver

$$h_b \leftarrow \max_{\ell \in \Phi_k} \{h_\ell\} .$$

Calculer α_ℓ, $\ell \in \Phi_k$ (le coût du chemin ℓ).

Poser:

$$d_i \leftarrow \begin{cases} \alpha_b - \alpha_\ell & \text{si } \alpha_b - \alpha_\ell \geq 0 \\ (\alpha_b - \alpha_\ell) h_\ell & \text{sinon} \end{cases} \Bigg\} \ell \neq b$$

$$d_\mu \leftarrow D_k'(\mu_k)\,(\mu_k - \alpha_b)$$

$$d_b \leftarrow D_k'(\mu_k)\, d_\mu - \sum_{\ell \neq b} d_\ell$$

Si $d=0$ terminer.

Normaliser d.

Etape 3: Trouver θ^*, $0 < \theta^* \leq 1$, qui minimise:

$$\sum_{ij \in A} \int_0^{f_{ij}+\theta \bar{d}_{ij}} C_{ij}(x)\,dx + \int_0^{\mu_k+\theta d_\mu} D_k(x)\,dx \qquad (4.11)$$

$$- (\mu_k + \theta d_\mu)\, D_k(\mu_k + \theta d_\mu)$$

où

$$\bar{d}_{ij} = \sum_{\ell \in \Phi_k} \delta_{ij,\ell}\, d_\ell \ ,$$

$$\delta_{ij,\ell} = \begin{cases} 1 & \text{si l'arc } ij \text{ fait partie du chemin } \ell \\ 0 & \text{sinon} \end{cases}$$

par une méthode appropriée (par exemple la méthode de la section dorée).

Etape 4: Faire

$$h_\ell \leftarrow h_\ell + \theta^* d \qquad \ell \in \Phi_k$$

$$\mu_k \leftarrow \mu_k + \theta^* d_\mu \ .$$

Si

$$\sum_{\ell \in \Phi_k} h_\ell = D_k(\mu_k) \ , \qquad \text{aller en 2}$$

Sinon, poser

$$j \leftarrow 1 \ , \quad \tilde{h}_b^j \leftarrow h_b \ .$$

4.1: Poser

$$\tilde{h}_b^{j+1} \leftarrow \tilde{h}_b^j - \sum_{\substack{\ell \in \Phi_k \\ \ell \neq b}} h_\ell + \tilde{h}_b^j - D_k(\mu_k)$$

Si $|\tilde{h}_b^{j+1} - \tilde{h}_b^j| > \varepsilon$, poser $j \leftarrow j+1$ et aller en 4.1

Sinon poser $h_b \leftarrow \tilde{h}_b^{j+1}$ et aller en 2

4.3 Une direction réalisable

Dans cette section, nous déterminons une direction de descente qui, replacée dans le contexte de la méthode des directions réalisables (section 2.1) et de la décomposition par paire O/D, permettra de résoudre le problème (P3).

On suppose que les fonctions de demande sont continuement dérivables et

strictement décroissantes.

Considérons le $k^{ième}$ sous-problème de (P3), $(SP3_k)$; soit $\{h_\ell, \ell \in \Phi_k; \mu_k\}$ une solution réalisable de ce sous-problème. On note par

$$\alpha_\ell = \frac{\partial F}{\partial h_\ell} = \sum_{ij \in A} a_{ij}^\ell \, C_{ij} \, (f_{ij})$$

le coût du chemin ℓ. Soit de plus

$$\alpha_b = \min_{\ell \in \Phi_k} \{\alpha_\ell\} .$$

On considère alors trois cas possibles:

a) $\mu_k > \alpha_b$; la direction est alors:

$$d_\mu = D_k' \, (\mu_k) \, (\mu_k - \alpha_b) < 0$$

$$\Theta d_b = D_k \, (\mu_k + \Theta d_\mu) - D_k \, (\mu_k) > 0 \quad \text{pour} \quad \Theta > 0$$

et $d_\ell = 0$ pour $\ell \neq b$.

Montrons d'abord que d est une direction réalisable. Puisque $\mu_k > 0$, les conditions de non-négativité sont sûrement respectées par la direction. D'autre part:

$$\sum_\ell h_\ell + \Theta d_\ell = \sum_{\ell \neq b} h_\ell + h_b + D_k \, (\mu_k + \Theta d_\mu) - D_k \, (\mu_k)$$

$$= \sum_\ell h_\ell - D_k \, (\mu_k) + D_k \, (\mu_k + \Theta d_\mu)$$

$$= D_k \, (\mu_k + \Theta d_\mu).$$

Il reste à montrer que d est une direction de descente, i.e.

$$\nabla F(h, \mu_k)' \, d < 0 .$$

On a:

$$\nabla F(h, \mu_k)' \, d = -\mu_k \, D_k'(\mu_k) \, d_\mu + \alpha_b \left[\frac{D_k(\mu_k + \theta d_\mu) - D_k(\mu_k)}{\theta} \right]$$

Or:

$$\lim_{\theta \to 0} \left(-\mu_k \, D_k'(\mu_k) d_\mu + \alpha_b \, [D_k(\mu_k + \theta d_\mu) - D_k(\mu_k)] / \theta \right)$$

$$= -\mu_k \, D_k'(\mu_k) d_\mu + \alpha_b \, D_k'(\mu_k) \, d_\mu$$

$$= D_k'(\mu_k) \, d_\mu \, (\alpha_b - \mu_k) < 0.$$

De par la définition de la limite, on sait qu'il existe un $\bar{\theta} > 0$ tel que:

pour tout θ : $0 < \theta \leq \bar{\theta}$ on a:

$$-\mu_k D_k'(\mu_k) d_\mu + \alpha_b [D_k(\mu_k + \theta d_\mu) - D_k(\mu_k)] / \theta < 0.$$

Donc, d est une direction de descente.

b) $\mu_k = \alpha_b$; on considère alors la direction suivante:

$$d_\mu = 0$$

$$d_b = D_k(\mu_k) - h_b.$$

$$d_\ell = -h_\ell.$$

Il est évident que d est une direction réalisable.
De plus,

$$\nabla F(h, \mu_k)' d = \alpha_b (D_k(\mu_k) - h_b) + \sum_{\ell \neq b} - \alpha_\ell \, h_\ell$$

$$= \alpha_b \, D_k(\mu_k) - \sum_\ell \alpha_\ell \, h_\ell \leq 0$$

$$\text{(puisque } \sum_\ell h_\ell = D_k(\mu_k)).$$

L'égalité est satisfaite lorsque

$$\alpha_b = \mu_k \quad \text{et} \quad \alpha_b = \alpha_\ell \qquad \text{pour tout} \quad \ell : h_\ell > 0 \ .$$

Mais alors les conditions de Kuhn-Tucker sont satisfaites et on est à l'optimum.

c) $\mu_k < \alpha_b$; considérons alors

$$d_\mu = D_k'(\mu_k)\ (\mu_k - \alpha_b) > 0$$

$$d_\ell = -h_\ell$$

$$\Theta d_b = D_k(\mu_k + \Theta d_\mu) - D_k(\mu_k) + \sum_{\ell \neq b} \Theta h_\ell$$

On a:

$$\sum_\ell h_\ell + \Theta d_\ell = \sum_\ell h_\ell - \sum_{\ell \neq b} \Theta h_\ell + D_k(\mu_k + \Theta d_\mu) - D_k(\mu_k) + \sum_{\ell \neq b} \Theta h_\ell$$

$$= D_k(\mu_k + \Theta d_\mu).$$

Cette direction est donc réalisable. D'autre part,

$$\nabla F(h, \mu_k)'\ d = -\mu_k D_k'(\mu_k)\ d_\mu - \sum_{\ell \neq b} \alpha_\ell\ h_\ell$$

$$+ \alpha_b\ [(D_k(\mu_k + \Theta d_\mu) - D_k(\mu_k))/\Theta + \sum_{\ell \neq b} h_\ell]$$

En prenant la limite:

$$\lim_{\Theta \to 0} \nabla F(h, \mu_k)'d = -\mu_k D_k'(\mu_k) d_\mu - \sum_{\ell \neq b} \alpha_\ell h_\ell + \alpha_b (D_k'(\mu_k) d_\mu + \sum_{\ell \neq b} h_\ell)$$

$$= D_k'(\mu_k) d_\mu (\alpha_b - \mu_k) - \sum_{\ell \neq b} \alpha_\ell h_\ell + \alpha_b \sum_{\ell \neq b} h_\ell < 0$$

$$\text{car} \quad \alpha_b = \min_\ell \alpha_\ell$$

On a donc une direction de descente.

CINQUIÈME CHAPITRE

UNE MÉTHODE DE RÉSOLUTION PAR DÉCOMPOSITION DE LA FONCTION OBJECTIF ET DES CONTRAINTES

5.1. Introduction

Il existe plusieurs méthodes efficaces pour résoudre le problème d'affectation du trafic avec demande fixe (voir [14]). La méthode présentée dans ce chapitre permet de résoudre le problème d'affectation avec demande variable en résolvant une suite de problèmes avec demande fixe. En effet, la fonction objectif de (P3) est séparable:

$$\text{Min } F = \sum_{ij \in A} \int_0^{f_{ij}} C_{ij}(x)dx + \sum_{k=1}^{n_p} \int_0^{\mu_k} D_k(x)dx - \mu_k D_k(\mu_k)$$

$$= N(f) + \zeta(\mu)$$

Il en est de même pour les contraintes. Ainsi, en fixant la valeur de μ on retrouve un problème avec demande fixe.

La méthode de Benders généralisée [6] permet d'effectuer un choix judicieux des valeurs de μ pour lesquelles on résoud (P3) afin de déterminer le point d'équilibre. Notons que cette méthode a déjà été appliquée au problème (P1) par Florian et Nguyen [4].

5.2. La méthode de Benders généralisée

Dans cette section, nous exposons la généralisation de l'algorithme de Benders dûe à Geoffrion [6].

Considérons le problème suivant:

$$\underset{x,y}{\text{Min}} \; f(x,y) \quad \text{s.a.} \quad G(x,y) \geq 0,\ x \in X,\ y \in Y \tag{5.1}$$

où G est un vecteur de fonctions de dimension m, $X \subseteq R^{n_1}$ et $Y \subseteq R^{n_2}$.

Supposons que lorsque y est fixé, le problème (5.1) est plus facile à résoudre que lorsque y varie; par exemple, pour y fixé, $f(x,y)$ et $G(x,y)$ pourraient être convexes, et ne pas posséder cette propriété lorsque y varie. Ainsi, en projetant le problème (5.1) sur l'axe y nous obtenons un problème équivalent, mais plus facile à résoudre:

$$\underset{y}{\text{Min}} \; v(y) \quad \text{s.a.} \quad y \in Y \cap V \tag{5.2}$$

où

$$v(y) = \inf_{x} f(x,y) \quad \text{s.a.} \quad G(x,y) \geq 0 \quad x \in X.$$

et

$$V = \{y \mid G(x,y) \geq 0 \quad \text{pour un} \quad x \in X\}$$

D'après nos hypothèses, il est beaucoup plus facile d'évaluer la fonction $v(y)$ (et donc de résoudre (5.2)) que de résoudre (5.1). Cependant, une difficulté majeure rencontrée lorsqu'on résoud (5.2) est d'identifier les $y \in V$; la méthode de Benders généralisée utilise la représentation duale de V en termes de l'intersection d'une suite d'ensembles qui contiennent V. En outre, on n'utilise pas directement la fonction $v(y)$, mais sa représentation duale.

Représentation duale de V

Geoffrion a démontré le théorème suivant:

Théorème: Soit X un ensemble convexe non-vide et supposons que G est

une fonction convexe sur X pour chaque valeur de $y \in Y$. Supposons de plus que l'ensemble $Z_y = \{Z \in R^m \mid G(x,y) \geq Z \text{ pour un } x \in X\}$ est fermé pour chaque $y \in Y$. Alors un $\bar{y} \in Y$ est aussi dans V ssi

$$\inf_{x \in X} \lambda^t G(x, \bar{y}) \geq 0 \qquad \text{pour tout } \lambda \in \Lambda \tag{5.3}$$

où

$$\Lambda = \{\lambda \in R^m \mid \lambda \geq 0 \text{ et } \sum_{i=1}^{m} \lambda_i = 1\}$$

Nous ne reproduirons pas la preuve de ce théorème ici. Notons simplement que les conditions d'application de ce théorème sont toujours satisfaites par le problème (P3).

Représentation duale de v

Sous certaines conditions qui sont toujours satisfaites dans le cas qui nous intéresse et que nous omettons ici, on a:

$$v(y) = \sup_{\mu \geq 0} \{\inf_{x \in X} f(x,y) + \mu^t G(x, y)\} \quad \text{pour tout } y \in Y \cap V \tag{5.4}$$

Ainsi, combinant (5.2), (5.3) et (5.4) on obtient le problème principal ("master problem") suivant:

$$\text{Min } y_0 \tag{5.5a}$$

$$\text{s.a.} \quad y_0 \geq \inf_{x \in X} \{f(x,y) + \mu^t G(x,y)\} \qquad \text{pour tout } \mu \geq 0 \tag{5.5b}$$

$$\inf_{x \in X} \{\lambda^t G(x,y)\} \geq 0 \qquad \text{pour tout } \lambda \in \Lambda \tag{5.5c}$$

qui est équivalent au problème (5.1).

Le problème (5.5) possède un très grand nombre de contraintes. Par consé-

quent, la stratégie de résolution à employer est la relaxation. Par cette stratégie, on résoud d'abord une version du problème (5.5) qui ne contient qu'un petit nombre de contraintes. Si la solution trouvée n'est pas réalisable pour le problème originel, on ajoute quelques contraintes, prises parmi celles qui sont le moins satisfaites. On termine lorsque la solution obtenue est "ε-optimale". Il faut donc élaborer une méthode permettant de vérifier si une solution donnée est réalisable ou non; dans ce dernier cas, on doit trouver une contrainte non satisfaite. On vérifiera si la solution est ε-optimale en calculant la différence entre une borne supérieure et une borne inférieure sur la valeur de la solution optimale.

Soit $(\hat{y}, \hat{y}_0)$ la solution optimale d'une version relaxée du problème (5.5). On veut vérifier si $(\hat{y},\hat{y}_0)$ satisfait (5.5b) et (5.5c) pour tous les $\mu \geq 0$ et tous les $\lambda \in \Lambda$. Par le théorème précédent et la définition de V, il est facile de voir que $\hat{y}$ satisfait les contraintes (5.5c) si et seulement si le problème suivant:

$$\min_{x \in X} \quad f(x,\hat{y}) \quad \text{s.a.} \quad G(x,\hat{y}) \geq 0 \tag{5.6}$$

possède une solution réalisable.

Supposons donc que le problème (5.6) possède une solution réalisable; la valeur de cette solution sera $v(\hat{y})$. Alors, par (5.4), $(\hat{y},\hat{y}_0)$ satisfait (5.5b) ssi

$$\hat{y}_0 \leq v(\hat{y}) \ .$$

Ainsi, le problème (5.6) permet de tester la réalisabilité d'une solution $(\hat{y},\hat{y}_0)$.

Définissons:

$$L^*(y,\mu) = \inf_{x\in X} \{f(x,y) + \mu^t G(x,y)\}, \qquad y\in Y,\ \mu \geq 0.$$

et

$$L_*(y,\lambda) = \inf_{x\in X} \{\lambda^t G(x,y)\} \qquad , \qquad y\in Y,\ \lambda\in \Lambda \ .$$

Nous pouvons maintenant présenter l'algorithme de Benders généralisé:

Algorithme B

Etape 1: Soit $\bar{y} \in Y \cap V$.

Résoudre le problème:

$$\min_{x\in X} f(x,\bar{y}) \quad \text{s.a.} \quad G(x,\bar{y}) \geq 0.$$

Soit $\bar{\mu}$ le vecteur de multiplicateurs optimaux (voir [6], section 5) associé à la solution optimale de ce problème.

Poser $p \leftarrow 1$, $q \leftarrow 0$, $\mu^1 \leftarrow \bar{\mu}$, $BSUP \leftarrow v(\bar{y})$

Etape 2: Résoudre:

$$\min_{y_0,y} \ y_0$$

$$\text{s.a.} \quad y_0 \geq L^*(y,\mu^j) \qquad j=1,\ldots,p \ ,$$

$$L_*(y,\lambda^j) \geq 0 \qquad j=1,\ldots,q \ .$$

Soit $(\hat{y},\hat{y}_0)$ la solution optimale de ce problème.

Si $BSUP - \hat{y}_0 \leq \varepsilon$ ($\varepsilon>0$ est la tolérance), terminer.

Etape 3: Résoudre:

$$\min_{x\in X} f(x,\hat{y}) \quad \text{s.a.} \quad G(x,\hat{y}) \geq 0$$

Aller en 3.1 si ce problème est réalisable, et en 3.2 sinon.

3.1: Si $v(\hat{y}) - \hat{y}_0 \leq \varepsilon$ terminer.

Sinon, déterminer un vecteur de multiplicateurs optimaux $\hat{\mu}$.

Poser $p \leftarrow p + 1$, $\mu^p \leftarrow \hat{\mu}$.

Si BSUP $> v(\hat{y})$ poser BSUP $\leftarrow v(\hat{y})$.

Aller en 2.

3.2: Trouver $\hat{\lambda} \in \Lambda$ tel que

$$\sup_{x \in X} \{\hat{\lambda}^t G(x,\hat{y})\} < 0$$

Poser $q \leftarrow q + 1$ et $\lambda^q \leftarrow \hat{\lambda}$.

Aller en 2.

5.3. Adaptation de la méthode de Benders généralisée au problème d'affectation du trafic avec demande variable (lorsque les inverses des fonctions de demande sont inconnues)

Dans cette section, nous allons élaborer une version de l'algorithme de Benders qui sera spécialisée pour notre problème. L'algorithme trouve le flot d'équilibre pour une paire O/D. Une application cyclique de cet algorithme (cf. section 2.2) permettra de résoudre (P3).

Notons que puisque les contraintes de (P3) sont toutes des contraintes d'égalité, la fonction $L_*(y,\lambda)$ n'a pas à être introduite ici.

Algorithme B1

Etape 1: Choisir $\hat{y}^0$.

Poser BSUP $\leftarrow \infty$ et $J \leftarrow 1$.

Etape 2: Poser $\bar{\mu} \leftarrow \hat{y}^{J-1}$

Résoudre le problème suivant:

$$\text{Min } F = \sum_{ij \in A} \int_0^{f_{ij}} C_{ij}(x)dx$$

$$\text{s.a.} \quad \sum_{\ell \Phi_k} h_\ell = D_k(\bar{\mu})$$

$$h_\ell \geq 0 \qquad (\ell \in \Phi_k)$$

$$f_{ij} = \sum_{\ell \in \Phi} a_{ij}^{\ell} \, h_\ell \quad .$$

Dénotons par $\bar{\mu}^J$ le coût à l'équilibre pour ce problème.

Poser:

$$\text{BSUP} \leftarrow \text{Min } \{\text{BSUP}, F + \int_0^{\bar{\mu}} D_k(x)dx - \bar{\mu} \, D_k(\bar{\mu})\}.$$

Etape 3: Résoudre le problème suivant:

$$\text{Min } y_0^J \qquad \text{(PP)}$$

$$\text{s.a. } L^*(y, \bar{\mu}^j) \leq y_0^J \quad , \qquad j=1,\ldots,J \; .$$

Notons par $(\hat{y}^J, \hat{y}_0^J)$ la solution optimale obtenue.

Si $\text{BSUP} - \hat{y}_0^J \leq \varepsilon_1$ ou $\bar{\mu}^J - \bar{\mu} \leq \varepsilon_2$ terminer

($\varepsilon_1 > 0$, $\varepsilon_2 > 0$ sont des tolérances pré-déterminées).

Sinon, poser $J \leftarrow J + 1$ et aller en 2.

(Nous verrons plus loin qu'il est possible de remplacer le test d'optimalité de l'étape 3 par un test plus simple).

Le problème d'optimisation de l'étape 2 est un problème d'affectation du trafic avec demande fixe et il peut donc être facilement solutionné à l'aide de la méthode du gradient réduit (voir [14]).

Il nous faut maintenant énoncer un algorithme permettant de résoudre le problème de l'étape 3. Explicitons d'abord les fonctions $L^*(y,\mu)$:

$$L^*(f,y,\bar{\mu}^j) = \min_{f_{ij}\geq 0} \sum_{ij\in A} \int_0^{f_{ij}} C_{ij}(x)dx + \int_0^y D_k(x)dx - yD_k(y)$$

$$+ \bar{\mu}^j (D_k(y) - \sum_{j\in\Phi_k} h_j)$$

$$= \min_{f_{ij}\geq 0} N(f) + \zeta_k(y) + \bar{\mu}^j (D_k(y) - B(f))$$

(où

$$N(f) = \sum_{ij\in A} \int_0^{f_{ij}} C_{ij}(x)dx$$

$$\zeta_k(y) = \int_0^y D_k(x)dx - y\, D_k(y)$$

et

$$B(f) = \sum_{j\in\Phi_k} h_j \ )$$

$$= \min_{f_{ij}\geq 0} N(f) + \bar{\mu}^j (D_k(\hat{y}^{j-1}) - B(f)) + \bar{\mu}^j (D_k(y) - D_k(\hat{y}^{j-1})) + \zeta_k(y)$$

$$= \min_{f_{ij}\geq 0} N(f) - \bar{\mu}^j D_k(\hat{y}^{j-1}) + \bar{\mu}^j D_k(y) + \zeta_k(y)$$

$$= N(\bar{f}) - \bar{\mu}^j D_k(\hat{y}^{j-1}) + \bar{\mu}^j D_k(y) + \zeta_k(y)$$

$$= s^j + \bar{\mu}^j D_k(y) + \zeta_k(y) = L^*(y,\bar{\mu}^j) .$$

Cette fonction est unimodale, et son minimum se situe en $\bar{\mu}^j$. En effet:

$$\frac{\partial L^*}{\partial y} = D_k'(y)\ (\bar{\mu}^j - y) \begin{cases} < 0 & \text{si } y<\bar{\mu}^j \\ > 0 & \text{si } y>\bar{\mu}^j \\ = 0 & \text{si } y=\bar{\mu}^j \end{cases} \qquad (5.7)$$

puisque la fonction de demande $D_k(y)$ est supposée strictement décroissante.

Supposons que $(\hat{y},\hat{y}_0)$ est la solution optimale du problème (PP) avec J-1 contraintes. La nouvelle contrainte n'est pas satisfaite par $(\hat{y},\hat{y}_0)$, sauf si l'on est à l'équilibre. En effet, supposons:

$$L^*(\hat{y},\bar{u}^J) \leq \hat{y}_0 \ .$$

Alors la solution optimale de (PP) lorsque l'on ajoute la $J^{ième}$ contrainte est évidemment:

$$\hat{y}' = \hat{y}$$

et $$\hat{y}'_0 = \hat{y}_0 \ .$$

D'où:

$$\bar{\mu}^{J+1} = \bar{\mu}^J \tag{5.8}$$

et alors on est exactement à l'optimum, car seul le point d'équilibre est tel que (5.8) est vérifié. Nous avons donc

$$\hat{y}_0 < L^* (\hat{y},\bar{\mu}^J) \tag{5.9}$$

et $$\hat{y}_0 \geq L^* (\hat{y},\bar{\mu}^j) \qquad , j=1,\ldots,J-1 \ . \tag{5.9a}$$

Ainsi, la solution optimale $(\hat{y}',\hat{y}'_0)$ de (PP) lorsque la $J^{ième}$ contrainte est ajoutée sera telle que

$$L^*(\hat{y}',\bar{\mu}^J) = \hat{y}'_0 \tag{5.10}$$

De plus, puisque les fonctions L^* sont unimodales (selon (5.7)), $\hat{y}'$ sera soit égal à $\bar{\mu}^J$ (puisque $\bar{\mu}^J$ est le minimum global de $L^*(y,\bar{u}^J)$) ou à un des J-1 points d'intersection de $L^*(y,\bar{\mu}^J)$ avec les $L^*(y,\bar{\mu}^j)$,

$j=1,\ldots,J-1$. D'autre part, par (5.10) la solution optimale $\hat{y}'$ sera atteinte en un point où la fonction a une valeur supérieure ou égale à celle des fonctions $L^*(y,\bar{\mu}^j)$, $j=1,\ldots,J-1$; en d'autres termes, la solution optimale $\hat{y}'$ est atteinte dans l'intervalle I sur lequel $L^*(y,\bar{\mu}^J)$ est maximale (voir figure 5.1). Il faut noter que par (5.9) et (5.9a), on a que $\hat{y}\in I$.

Dénotons par Z^j, $j=1,\ldots,J-1$ les $J-1$ points d'intersection de $L^*(y,\bar{\mu}^J)$ avec les $L^*(y,\bar{\mu}^j)$. Nous formalisons les considérations précédentes par la proposition suivante:

Proposition: Supposons que la solution optimale de (PP) avec J contraintes ne soit pas en $\bar{\mu}^J$, i.e. qu'il existe un j, $1 \le j \le J-1$, tel que

$$L^*(\bar{\mu}^J,\bar{\mu}^J) < L^*(\bar{\mu}^J,\bar{\mu}^j).$$

Soit $INT = \{j \mid L^*(Z^j,\bar{\mu}^J) > \hat{y}_0\}$, et soit $\bar{Z}$ tel que

$$L^*(\bar{Z},\bar{\mu}^J) = \min_{j\in INT} \{L^*(Z^j,\bar{\mu}^J)\}.$$

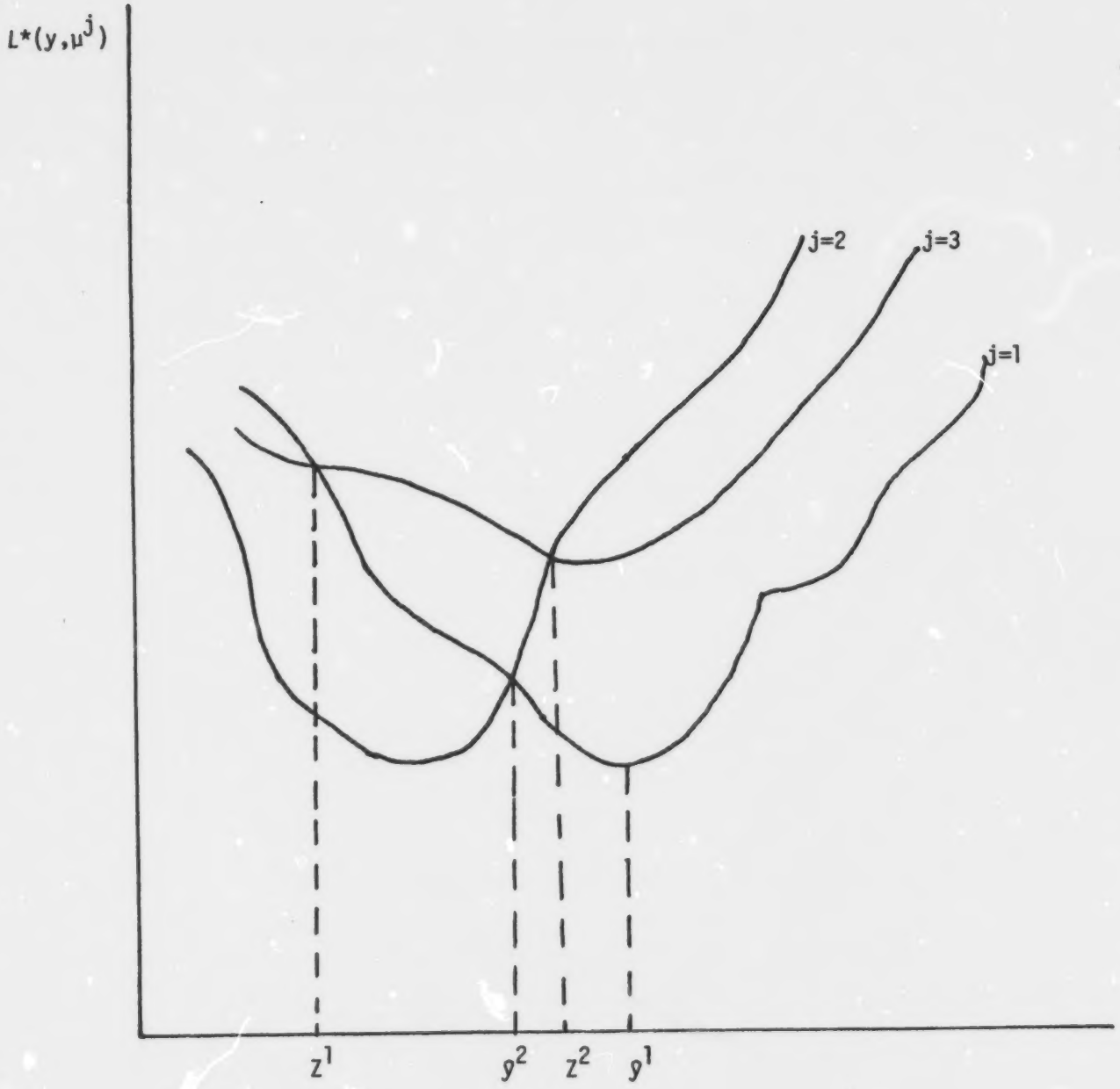
Alors $(\hat{y}',\hat{y}_0') = (\bar{Z},L^*(\bar{Z},\bar{\mu}^J))$ est la solution optimale de (PP).

L'inconvénient inhérent à toute méthode de résolution basée sur la proposition précédente semble être l'obligation de calculer à chaque étape tous les points d'intersection Z^j, ce qui risque d'être long et coûteux. Or, il est facile de trouver une procédure qui évite le calcul de tous les Z^j.

En effet, on sait que Z^j est tel que

$$L^*(Z^j,\bar{\mu}^j) = L^*(Z^j,\bar{\mu}^J),$$

c'est-à-dire:



Notes: $z^2 = \hat{y}^3$

$I = |z^1, z^2|$;

le nouvel optimum, $\hat{y}^3$, est dans l'intervalle I.

Figure 5.1: Résolution du problème (PP)

$$s^j + \bar{\mu}^j D_k(Z^j) + \zeta_k(Z^j) = s^J + \bar{\mu}^J D_k(Z^j) + \zeta_k(Z^j)$$

et alors:

$$D_k(Z^j) = (s^J - s^j) \,/\, (\bar{\mu}^j - \bar{\mu}^J) \tag{5.11}$$

On suppose $\bar{Z} \neq \bar{\mu}^J$. Alors par (5.7), (5.9) et la proposition précédente, $\bar{Z}$ doit être égal à Z^{j_1} ou Z^{j_2}, où

$$Z^{j_1} = \max \{0,\ Z^j : Z^j < \hat{y}\} \tag{5.12}$$

$$Z^{j_2} = \min \{\infty,\ Z^j : Z^j > \hat{y}\} \tag{5.13}$$

Puisque $D_k(x)$ est strictement décroissante, on peut réécrire (5.12) et (5.13) ainsi:

Z^{j_1} est tel que $D_k(Z^{j_1}) = \min\limits_j \{D_k(0), D_k(Z^j) : D_k(Z^j) > D_k(\hat{y})\}$

et Z^{j_2} est tel que $D_k(Z^{j_2}) = \max\limits_j \{0, D_k(Z^j) : D_k(Z^j) < D_k(\hat{y})\}$.

Les $D_k(Z^j)$ sont évalués à l'aide de l'expression (5.11). On pose:

$$D_k(\tilde{Z}^1) = \min \{D_k(Z^{j_1}),\ D_k(Z^{j_2})\}$$

et

$$D_k(\tilde{Z}^2) = \max \{D_k(Z^{j_1}),\ D_k(Z^{j_2})\}\ .$$

Enfin, si $D_k(\bar{\mu}^J) < D_k(\tilde{Z}^1)$ alors $\bar{Z} = \tilde{Z}^1$, sinon $\bar{Z} = \tilde{Z}^2$, et ceci par décroissance de D_k, par (5.7) et par la proposition précédente (plus précisément, par le fait que $\bar{\mu}^J \notin I$, alors que $\tilde{Z}^1 \in I$ et $\tilde{Z}^2 \in I$, ce qui implique que l'on a une des deux situations suivantes:

$$\bar{\mu}^J > \tilde{Z}^1 > \tilde{Z}^2$$

ou

$$\bar{\mu}^J < \tilde{Z}^2 < \tilde{Z}^1).$$

On peut finalement déterminer explicitement la valeur de $\bar{Z}$ en trouvant la racine de la fonction

$$D_k(Z) - (\bar{s} - s^J) / (\bar{\mu}^J - \bar{\mu}) . \qquad (5.14)$$

La procédure est donc la suivante:

Algorithme B2

Etape 1: S'il existe un j, $1 \leq j \leq J-1$, tel que

$$L^*(\bar{\mu}^J, \bar{\mu}^j) > L^*(\bar{\mu}^J, \bar{\mu}^J)$$

aller en 2.

Sinon, poser:

$$\hat{y}^J \leftarrow \bar{\mu}^J, \hat{y}_0^J \leftarrow L^*(\bar{\mu}^J, \bar{\mu}^J) \quad \text{et terminer}$$

Etape 2: Calculer:

$$D(Z^j) \leftarrow (s^J - s^j) / (\bar{\mu}^j - \bar{\mu}^J) \qquad j=1,\ldots,J-1$$

Soient

$$D(Z^{j_1}) \leftarrow \min_j \{D_k(0), D(Z^j) : D(Z^j) > D_k(\hat{y}^{J-1})\}$$

et

$$D(Z^{j_2}) \leftarrow \max_j \{0, D(Z^j) : D(Z^j) < D_k(\hat{y}^{J-1})\} .$$

Soient

$$D(\tilde{Z}^1) \leftarrow \min \{D(Z^{j_1}), D(Z^{j_2})\}$$

et

$$D(\tilde{Z}^2) \leftarrow \max \{D(Z^{j_1}), D(Z^{j_2})\}$$

Si $D_k(\bar{\mu}^J) < D(\tilde{Z}^1)$

alors poser $D(\bar{Z}) \leftarrow D(\tilde{Z}^1)$

sinon poser $D(\bar{Z}) \leftarrow D(\tilde{Z}^2)$

Trouver $\bar{Z}$ la racine de

$$D_k(Z) - (\bar{s} - s^J) / (\bar{\mu}^j - \bar{\mu})$$

(où $\bar{s}$ et $\bar{\mu}$ sont les s^j et $\bar{\mu}^j$ qui correspondent à $\bar{Z}$).

Poser $\hat{y}^J \leftarrow \bar{Z}$

et $\hat{y}_0^J \leftarrow L^*(\bar{Z}, \bar{\mu}^J)$.

Remarquons qu'il n'est pas nécessaire de calculer la valeur de $\bar{Z}$; en effet, à l'étape 3 de l'algorithme B1, le critère d'arrêt peut être remplacé par:

$$\text{si } |D_k(\hat{y}^J) - D_k(\hat{y}^{J-1})| \leq \varepsilon_3 \text{ , terminer.}$$

Ainsi, la valeur de $D_k(\bar{Z})$ est suffisante, et on n'a à déterminer explicitement aucun des points d'intersection Z^j.

La méthode que nous avons présentée est identique à celle proposée dans [4]. En effet, il suffit de remplacer, dans les algorithmes de Florian et Nguyen [4], l'intégrale de l'inverse de la fonction de demande par l'expression équivalente (proposée au premier chapitre) pour obtenir les algorithmes B1 et B2. Si l'on utilise le critère d'arrêt (5.15), il n'est pas nécessaire de connaître la valeur exacte des coûts de transport. La valeur optimale de la fonction-objectif de (PP), que l'on peut calculer analytiquement et qui donne la demande $D_k(\hat{y}^J)$, est donc suffisante. Par contre, lorsque l'on résoud le problème (P1), la valeur de la demande g^J est donnée par la solution optimale du problème principal; on doit donc résoudre un problème de minimisation unidimensionnel. Ainsi, la seule dif-

férence entre notre algorithme et celui présenté dans [4] est que nous pouvons résoudre analytiquement le problème principal.

5.4. Note sur la résolution du problème à l'étape 2 de l'algorithme B1: génération progressive des chemins dans l'algorithme B1

A l'étape 2 de l'algorithme de Benders généralisé (B1) on a un problème avec demande fixe qui ne comporte qu'une seule paire O/D à résoudre; pour obtenir la solution optimale de ce problème, il faut donc considérer l'ensemble de tous les chemins reliant la $k^{ième}$ paire O/D. C'est donc dire qu'on ne peut pas, contrairement à ce que suggère l'algorithme G1 (section 2.3), résoudre le problème avec demande variable sur toutes les paires O/D avant de réviser l'ensemble des chemins disponibles; il faut faire cette révision lors de la résolution du problème avec demande fixe défini à l'étape 2.

SIXIEME CHAPITRE

ANALYSE DES RESULTATS OBTENUS

6.1. Description des programmes

Tous les algorithmes étudiés dans ce mémoire ont été programmés en Fortran sur l'ordinateur CDC 173 du Centre de calcul de l'Université de Montréal.

Le tableau 6.1 présente une liste des programmes utilisés. Nous ne discuterons pas ici des programmes FW et GR qui supposent l'existence des inverses des fonctions de demande et qui sont largement commentés dans [14].

6.1.1. Structure des données et espace-mémoire

Dans cette section, nous décrivons la structure de données utilisée pour représenter le réseau et nous déterminons l'espace-mémoire nécessaire à l'exécution de chacun des programmes. Cette analyse est basée sur celle faite dans [14].

Les arcs constituant le réseau sont représentés en mémoire de la façon suivante. Les arcs sont numérotés de 1 à m et les sommets de 1 à n. Chaque arc est décrit par son sommet initial et son sommet final. On suppose que les arcs sont triés selon l'ordre croissant des numéros des sommets initiaux; on enregistre le numéro du sommet terminal de chacun des arcs dans un vecteur de dimension m; on enregistre l'adresse du premier des arcs (x,y) (pour chacun des x) dans le $x^{ième}$ élément d'un vecteur de dimension n. En plus d'économiser de l'espace-mémoire (puisque le nombre de sommets est généralement beaucoup plus petit que le nombre

Nom du programme	Problème solutionné	Algorithme utilisé
FW*	P1	Frank et Wolfe
FW2	P2	Frank et Wolfe (linéarisation des fonctions de demande)
GR*	P1	gradient réduit
LINDYN	P1	linéarisation progressive, version 1
LINDYN2	P1	linéarisation progressive version 2
GRG	P3	gradient réduit généralisé
DIRRES	P3	direction réalisable
ETABD	P3	Benders généralisé

(Tous les programmes sont des modifications de programmes écrits par Sang Nguyen [14])

Tableau 6.1 - Programmes utilisés

* Ces programmes ne peuvent être utilisés que lorsque les inverses des fonctions de demande sont connues.

d'arcs) cette représentation facilite grandement la recherche des chemins les plus courts.

L'espace-mémoire utilisé se divise en quatre sections:

a) l'espace nécessaire pour codifier le réseau;

b) l'espace-mémoire nécessaire pour conserver les informations sur les coûts et les demandes de déplacement interzonal;

c) l'espace-mémoire pour la codification des fonctions de demande et des coûts d'utilisation des arcs;

et finalement,

d) la zone de travail

On a déjà discuté de la codification du réseau. L'espace utilisé par les informations des types (a) et (b) est le même dans tous les programmes.

Les programmes étudiés se divisent en deux catégories qui se distinguent par l'importance de la zone de travail utilisée, d'une part, et de l'espace nécessaire à la codification des fonctions de demande, d'autre part. Dans la première catégorie, on retrouve l'algorithme de Frank et Wolfe (programme FW2); dans la seconde, on a tous les programmes basés sur la décomposition par paire origine-destination, soient les programmes LINDYN, LINDYN2, GRG, DIRRES et ETABD.

La taille de la zone de travail dans le programme implantant la méthode de Frank et Wolfe est de l'ordre de 3 (n+m), où n est le nombre de sommets et m le nombre d'arcs dans le réseau. En effet, il nous faut:

i) 3 tableaux de dimension m, contenant le flot sur chacun des arcs, le

coût d'utilisation de chacun de ces arcs, et finalement la nouvelle solution trouvée à l'étape courante;

ii) 3 tableaux de dimension n utilisés lors de la recherche des chemins les plus courts.

Les algorithmes de la seconde catégorie, qui utilisent tous la méthode de décomposition par paire origine-destination, exigent un espace de travail de l'ordre de $2m + 3n + 3n_p + 2q + q\ell$, où n_p est le nombre de paires O/D, q est le nombre maximal de chemins utilisés (dans leur version actuelle, les programmes prévoient 5 chemins pour chacune des paires O/D; on a donc $q = 5n_p$), et enfin ℓ est le nombre moyen d'arcs prévu dans chacun des chemins. Le programme ETABD (implantant la méthode de Benders généralisée) exige un espace supplémentaire de l'ordre de $3k$, où k est le nombre maximal de contraintes dans le problème (PP).

Les 3 tableaux de dimension n ont les mêmes fonctions que ceux du programme FW2; les 2 tableaux de dimension m sont utilisés pour conserver les flots sur les arcs et les coûts associés à l'utilisation de ces arcs. Les 3 tableaux de dimension n_p servent à conserver le numéro du noeud d'origine et celui du noeud de destination de chacune des paires O/D, ainsi que le nombre de chemins utilisés; les 2 tableaux de dimension q contiennent le flot sur chacun des chemins et le nombre d'arcs composant chacun des chemins utilisés. Les q tableaux de dimension ℓ contiennent la description des chemins utilisés en termes d'arcs.

Etant donné l'importance de la zone de travail dans les algorithmes utilisant la décomposition par paires O/D, on doit ici recourir à l'utilisation de la mémoire auxiliaire.

La codification des fonctions de demande exige, dans le programme FW2, un espace-mémoire de l'ordre de

$$3 \sum_{k=1}^{n_p} m_k$$

où m_k est le nombre de morceaux composant l'approximation linéaire de la fonction de demande de la $k^{ième}$ paire O/D. D'autre part, les programmes de la seconde catégorie exigent ici un espace-mémoire de l'ordre de

$$\sum_{k=1}^{n_p} r_k$$

où r_k est le nombre de paramètres caractérisant la $k^{ième}$ fonction de demande.

6.1.2. Description générale des programmes

Une copie des programmes implantant les algorithmes dont nous avons traité dans les troisième, quatrième et cinquième chapitres (soient les programmes LINDYN, LYNDYN2, DIRRES, GRG et ETABD) est donnée en appendice.

Dans cette section, nous décrirons brièvement le rôle des sous-routines les plus importantes de chacun de ces programmes.

La sous-routine TRAFIC, en plus de trouver la solution initiale, effectue la décomposition par paires O/D (section 2.2), et, sauf pour ce qui est du programme ETABD, implante la méthode de génération progressive des chemins utilisés (section 2.3) via la sous-routine REVPTH.

La sous-routine EQUIL trouve le point d'équilibre pour une paire O/D selon la méthode propre à l'algorithme programmé.

La sous-routine GSM trouve le pas optimal à l'aide de la méthode de la section dorée [12].

La sous-routine MINTRE effectue la recherche du chemin le plus court selon la méthode de Dijkstra [3]; les étiquettes temporaires sont conservées dans une structure de "heap" [5,10].

Dans les programmes LINDYN et LINDYN2, la sous-routine MONEQU trouve la fonction linéaire qui approxime l'inverse de la fonction de demande.

Enfin, dans le programme ETABD, la sous-routine P2 trouve la solution du problème principal (problème (PP)) tandis que la sous-routine EQFIX trouve la solution du problème avec demande fixe défini à l'étape 2 de l'algorithme B1 (section 5.3); la procédure GB coordonne le travail de ces deux sous-routines et détermine le point d'équilibre d'une paire origine-destination.

6.2. Résultats obtenus

Dans cette section, nous présentons d'abord les données utilisées pour tester les programmes, nous discutons brièvement des résultats obtenus, puis nous classifions les différents algorithmes selon leur temps d'exécution.

6.2.1. Données utilisées

La figure 6-1 illustre le réseau utilisé. Ce réseau est constitué de 13 sommets et 19 arcs, et comporte 4 paires origine-destination.

Aucune capacité n'est imposée sur les arcs. Le coût marginal d'utilisation des arcs est défini comme suit:

$$c_{ij}(f_{ij}) = a_{ij} f_{ij} + b_{ij}$$

où f_{ij} est le flot sur l'arc ij; les valeurs des constantes a_{ij} et b_{ij} sont inscrites dans le tableau 6.2

Les fonctions de demande utilisées sont de la forme

$$g = 1000\, e^{-\theta\mu} \qquad (6.1)$$

dont l'inverse est

$$\mu = -\ell n(g/1000)/\theta. \qquad (6.2)$$

Les valeurs du paramètre θ sont présentées dans le tableau 6.3; elles ont été calculées de façon à ce que les points d'équilibre soient tels que définis dans ce tableau.

Le réseau et les coûts d'utilisation des arcs proviennent de [15].

6.2.2. Discussion des résultats

Les tableaux 6.4 et 6.5 présentent le temps d'exécution de chacun des programmes, ainsi que les valeurs optimales des fonctions-objectif. Les solutions d'équilibre (μ^*, g^*) obtenues sont identiques (aux erreurs d'arrondi près) à celle que l'on a calculée de façon théorique (voir la section 6.2.1 et le tableau 6.3).

On remarque que les programmes étudiés se divisent en deux catégories. Dans la première, la valeur optimale de la fonction-objectif est de l'ordre de -5×10^4 ; on y retrouve tous les programmes qui résolvent (P1), c'est-à-dire GR, LINDYN et LINDYN2. La seconde catégorie regroupe les programmes qui solutionnent le problème (P3) et la valeur optimale de la fonction objectif est de l'ordre de -2×10^5 . Cette différence s'explique

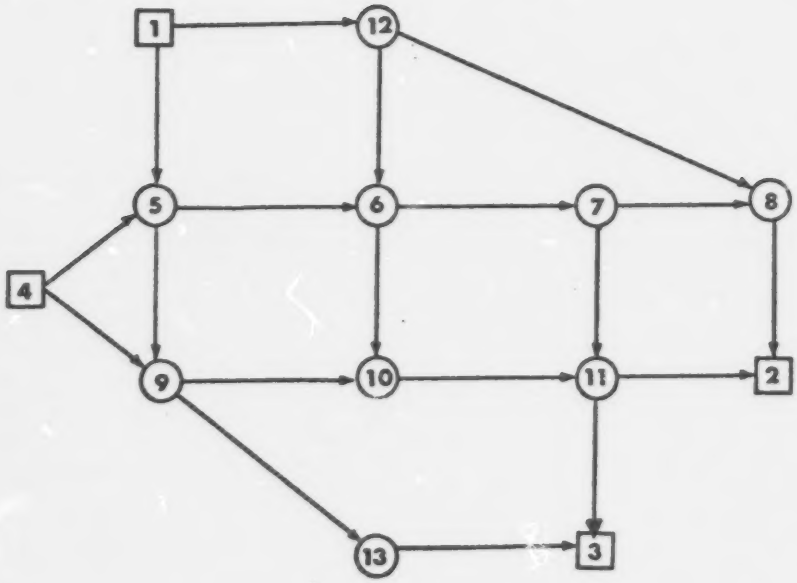


Figure 6.1 - Réseau utilisé

Arc	Origine	Destination	a_{ij}	b_{ij}
1	1	5	0.0125	7
2	1	12	0.01	9
3	4	5	0.01	9
4	4	9	0.005	12
5	5	6	0.0075	3
6	5	9	0.0075	9
7	6	7	0.0125	5
8	6	10	0.005	13
9	7	8	0.0125	5
10	7	11	0.0125	9
11	8	2	0.0125	9
12	9	10	0.005	10
13	9	13	0.005	9
14	10	11	0.0025	6
15	11	2	0.005	9
16	11	3	0.01	8
17	12	6	0.0025	7
18	12	8	0.01	14
19	13	3	0.01	11

Tableau 6.2 - Description des arcs et des coûts d'utilisation

Paire O/D	Origine	Destination	Paramètre	Point d'équilibre (μ^*,g^*)
1	1	2	0.0193	(47.48,400)
2	1	2	0.004	(55.76,800)
3	4	3	0.0108	(47.3,600)
4	4	3	0.0367	(43.85,200)

Tableau 6.3 - Description des paires origine-destination et des paramètres des fonctions de demande

par le fait que les fonctions-objectif de (P1) et (P3) sont égales à une constante près (cf. section 1.2.2.2.).

On constate de plus que, en utilisant le même critère, le premier groupe de programmes peut être subdivisé: cela est dû au fait que les programmes LINDYN et LINDYN2 utilisent des approximations linéaires des inverses des fonctions de demande alors que le programme GR utilise directement ces fonctions inverses.

Les remarques précédentes sont confirmées par le fait que les valeurs de

$$\sum_{ij \in A} \int_0^{f_{ij}} C_{ij}(x)dx$$

ne diffèrent que très peu d'un programme à l'autre.

6.2.3. Classification des algorithmes

Le réseau utilisé étant très petit et ne réflétant aucune donnée réelle, il est très hasardeux de tenter de tirer des conclusions sur l'efficacité relative des algorithmes en se basant sur les temps d'exécution.

Cependant, on peut tout de même faire quelques remarques générales. En premier lieu, il faut noter que le calcul du pas optimal exige la presque totalité du temps d'exécution, et ce, dans tous les programmes sauf ETABD.

La méthode de Benders généralisée (programme ETABD) est de loin la plus efficace. Il faut remarquer que dans ce programme le temps de calcul du pas optimal ne représente que 33% du temps total d'exécution. Contrairement aux autres programmes, ETABD ne calcule aucune direction qui puisse solutionner un problème avec demande variable: en effet, la méthode consiste

Programme	Temps total d'exécution	Temps de calcul de $\lambda \pm 0.0001$	Valeur de la fonction objectif	$\sum \int C_{ij}(x)dx$
GR	2.37	1.64	-4.66022×10^4	8.38059×10^4
LINDYN	1.65	0.89	-2.87720×10^4	8.567401×10^4
LINDYN2	1.62	0.90	-2.87720×10^4	8.567401×10^4
GRG	4.11	3.28	-2.98418×10^5	8.429308×10^4
DIRRES	1.66	1.05	-2.98557×10^5	8.625279×10^4
ETABD	0.63	0.21	-2.98437×10^5	8.53952×10^4

Tableau 6.4 - Comparaison des algorithmes utilisant la méthode de décomposition par paires O/D; critère d'arrêt: coûts des chemins utilisés égaux entre eux à 5% près.

Programme	Temps total d'exécution (sec.)	Temps de calcul de $\lambda \pm 0.0001$ (sec.)	Valeur de la fonction objectif	$\sum \int C_{ij}(x)dx$
GR	4.55	3.54	-4.68695×10^4	8.509757×10^4
LINDYN	2.85	1.95	-5.65157×10^4	8.595829×10^4
LINDYN2	2.86	1.97	-5.65157×10^4	8.595829×10^4
GRG	6.54	5.41	-2.98490×10^5	8.514595×10^4
DIRRES	21.71	19.83	-2.98657×10^5	8.529142×10^4
ETABD	0.68	0.27	-2.98477×10^5	8.538773×10^4

Tableau 6.5 - Comparaison des algorithmes utilisant la méthode de décomposition par paires O/D; critère d'arrêt: coûts des chemins utilisés égaux entre eux à 1% près.

essentiellement à résoudre une suite de problèmes avec demande fixe, et c'est probablement ce qui explique sa très grande efficacité.

Viennent ensuite les deux programmes de linéarisation progressive; on ne peut pas déceler de différence significative entre ces deux algorithmes. On sait cependant que la seconde version de l'algorithme (programme LINDYN2) est plus efficace que la première (voir section 3.2), ce qu'on pourrait sûrement vérifier en solutionnant un problème réel à l'aide de ces deux algorithmes.

Viennent enfin l'algorithme du gradient réduit généralisé, puis la direction réalisable définie à la section 4.3.

Remarquons que l'algorithme du gradient réduit (que l'on ne peut utiliser que lorsque les inverses des fonctions de demande sont connues) est plus efficace que sa généralisation au cas de contraintes non-linéaires. En effet, le gradient réduit exige en général moins d'itérations que le gradient réduit généralisé (140 et 214 itérations respectivement, lorsqu'on exige que les coûts d'utilisation des chemins soient égaux à 1% près), le temps moyen de calcul du pas optimal étant le même dans les deux cas. On pouvait s'attendre à cela puisque, lorsque le pas optimal dans la direction GRG a été calculé, on modifie la valeur de la variable de base afin de respecter la contrainte:

$$\sum_{\ell \in \phi_k} h_\ell = D_k(\mu_k)$$

tandis que la direction du gradient réduit est réalisable.

Le programme implantant la direction réalisable (section 4.3) n'est pas

efficace. Le point faible de la direction semble être le cas où la valeur du coût de déplacement interzonal μ_k est inférieure au coût α_b du chemin le plus court (cas (c), section 4.3) lorsque plusieurs chemins sont utilisés. Dans cette circonstance, on essaie d'ajuster à la fois le coût μ_k de déplacement interzonal (donc la demande g_k) et les flots de tous les chemins utilisés. Or, il peut arriver que la direction d_b de la variable associée au chemin à coût minimum soit positive. L'écart entre μ_k et α_b diminue alors beaucoup moins vite que dans le cas (a), puisque dans ce dernier cas d_b et d_{μ_k} sont de sens opposés. Il faut noter, de plus, que dans ce programme la direction n'a pas été normalisée, c'est-à-dire que la valeur optimale θ^* du pas n'est pas nécessairement comprise entre 0 et 1.

Les deux programmes implantant la méthode de Frank et Wolfe peuvent difficilement être comparés avec les précédents. En effet, l'efficacité de l'algorithme de Frank et Wolfe diminue très rapidement lorsqu'on approche de la solution optimale; pour cette raison, on a arrêté l'exécution du programme après 25 itérations, la valeur de la fonction objectif diminuant de façon négligeable à la $25^{ième}$ itération. La méthode de Frank et Wolfe ne donne pas d'information sur les coûts des chemins utilisés; on ne peut donc utiliser le critère d'arrêt qui a été appliqué aux programmes précédents, et toute comparaison avec ces programmes serait sans signification.

Mais on peut tout de même comparer les deux versions de l'algorithme de Frank et Wolfe entre elles; ces résultats sont présentés dans le tableau 6.6.

Les approximations linéaires des fonctions de demande ont été obtenues à l'aide de l'algorithme LIN1 (section 1.2.2.1.) et sont décrites dans le tableau 6.7; la différence maximale sur un segment [A,B] est définie comme suit:

$$\max_{\mu \in [A,B]} |\mu - L_k^{-1}(D_k(\mu))|$$

où D_k est la fonction de demande de la $k^{ième}$ paire O/D et L_k^{-1} est l'inverse de la fonction linéaire qui approxime D_k sur [A,B].

L'approximation linéaire utilisée est très précise et les solutions optimales obtenues avec les deux programmes sont à toutes fins pratiques identiques. La différence entre les valeurs optimales des fonctions-objectif tient évidemment au fait que ces deux fonctions sont différentes: dans un cas on emploie les intégrales des fonctions de demande et dans l'autre les intégrales des approximations linéaires.

La différence dans les temps d'exécution provient essentiellement de la difficulté d'évaluation de l'intégrale d'une fonction linéaire par morceaux (voir l'équation (1.6), section 1.2.2.1.); en effet, les évaluations des intégrales des inverses des fonctions de demande exigent 3.59 secondes dans FW2 et 0.46 seconde dans FW.

Programme	Temps total d'exécution (secondes)	Temps de calcul de $\lambda \pm 0.00001$ (secondes)	Valeur de la fonction-objectif
FW	2.09	1.59	-2.96995×10^5
FW2	5.51	4.75	-2.27453×10^5

Tableau 6.6 - Temps d'exécution de l'algorithme de Frank et Wolfe

Paire O/D	Borne inférieure	Borne supérieure ±0.1	Nombre de segments	Différence maximale
1	0	150	10	0.55
2	0	800	30	0.36
3	0	300	15	0.54
4	0	150	15	0.46

Tableau 6.7 - Caractéristiques des approximations linéaires utilisées

CONCLUSION

La recherche du flot d'équilibre dans un réseau routier urbain lorsque la demande de déplacement est variable a été formulé comme un problème de programmation convexe à contraintes linéaires par Beckmann *et al.* Cette formulation exige la connaissance a priori des inverses des fonctions de demande.

Or, il est parfois fort difficile d'obtenir ces fonctions inverses. Dans ce mémoire, nous avons étudié le problème de l'affectation du trafic routier avec demande variable lorsque les inverses des fonctions de demande ne sont pas connues.

Nous avons d'abord étudié une approche basée sur l'approximation linéaire des fonctions de demande. Nous avons appliqué le concept élaboré par Geoffrion pour la linéarisation des fonctions-objectif. Nous avons aussi élaboré un nouvel algorithme basé sur l'approximation linéaire progressive des fonctions de demande; cet algorithme permet de résoudre le problème formulé par Beckmann *et al.* même lorsque les inverses ne sont pas connues.

Nous avons ensuite élaboré un modèle mathématique qui n'utilise par les inverses des fonctions de demande; ce modèle s'inspire de celui de Beckmann *et al.*; c'est un problème de programmation mathématique non-convexe. Nous avons démontré l'équivalence entre les deux modèles. Nous avons finalement suggéré plusieurs algorithmes de résolution pour ce nouveau problème; ces algorithmes sont basés sur les méthodes des directions réalisables, du gradient réduit généralisé et de Benders.

Ainsi, ce mémoire constitue également un tour d'horizon des principaux algorithmes de programmation non-linéaire.

Tous les algorithmes élaborés ont été programmés en Fortran et testés sur l'ordinateur de l'Université de Montréal. Les résultats obtenus permettent d'effectuer une classification grossière des algorithmes.

La méthode de Benders est la plus efficace; cet algorithme est environ quatre fois plus efficace que l'algorithme basé sur la linéarisation progressive des fonctions de demande. Vient ensuite l'algorithme du gradient réduit généralisé qui est environ deux fois moins rapide que la linéarisation progressive. Finalement, la direction réalisable que nous avons élaborée est la moins efficace; elle est au fait inutilisable en pratique.

L'affectation du trafic est une étape importante dans le processus de la planification des réseaux de transport urbains. Ce mémoire répond à un besoin réel dans ce domaine puisque les algorithmes que nous présentons permettent d'effectuer l'affectation du trafic même lorsque les inverses des fonctions de demande ne sont pas connues.

APPENDICE A1

CONSERVATION DES CHEMINS A L'AIDE DE PIVOTS

A1.1. Introduction

Lorsque l'on résoud le problème d'affectation du trafic par une méthode basée sur la décomposition par paires origine-destination, il est nécessaire de conserver la description, en termes d'arcs, de chacun des chemins utilisés.

Or, si par exemple, une paire origine-destination utilise en moyenne cinq chemins, un problème de taille respectable compte alors environ 50 000 chemins. Ce très grand nombre de chemins peut marquer une limite quant à l'utilisation des méthodes basées sur la génération des chemins. D'où la nécessité de trouver une technique permettant d'enregistrer ces chemins de la façon la plus compacte possible.

La méthode qui est présentée ici est basée sur le concept de pivots. On construit une arborescence des chemins les plus courts à partir de chacune des origines. Pour retracer un chemin h entre une origine o_k et une destination d_ℓ données, il suffit alors de retracer les arcs qui diffèrent entre ceux qui composent le chemin h et ceux qui sont dans l'arborescence associée à l'origine o_k.

A1.2. Terminologie et notation

L'arborescence des chemins les plus courts est représentée par une fonction P(i) qui, au noeud i, associe le sommet précédent i dans l'arborescence (c'est-à-dire que l'arc (P(i),i) est dans l'arborescence). Le chemin h est représenté par la fonction CH(i) selon le même principe.

Un pivot est caractérisé par une paire ordonnée d'arcs: un arc entrant

(q_1,q_2) et un arc sortant (n_1,n_2). L'arc entrant est un arc de h, mais il n'est pas dans l'arborescence, et inversement pour l'arc sortant.

A1.3. Procédure

Considérons un chemin h que l'on veut ajouter entre l'origine o_k et la destination d_ℓ. On note par P_k la fonction représentant l'arborescence associée à l'origine o_k. La procédure est alors la suivante:

Algorithme P1

Etape 1: Poser $i \leftarrow d_\ell$.

Etape 2: Si $CH(i) = P_k(i)$ aller en 3.
Sinon, l'arc sortant est $(P_k(i),i)$ et l'arc entrant est $(CH(i),i)$.

Etape 3: Poser $i \leftarrow CH(i)$.
Si $i = o_k$, terminer, sinon aller en 2.

En fait, il n'est pas nécessaire de retenir l'arc entrant et l'arc sortant; il suffit de garder i et $CH(i)$ pour chaque i tel que $P_k(i) \neq CH(i)$.

Considérons, par exemple, l'arborescence représentée par la figure A1.1 et supposons que l'on veuille codifier le chemin représenté par la figure A1.2. L'information qu'il faudra alors conserver est $i=10$ et $CH(i)=3$, puisque, des arcs forment ce nouveau chemin seul l'arc $(3,10) \equiv (3,d_\ell)$ n'est pas dans l'arborescence associée à l'origine o_k.

Notons aussi que dans le pire des cas (qui surviendra assez fréquemment), cette méthode requiert autant d'informations que la méthode habituelle consistant à conserver la liste de tous les arcs constituant le chemin h.

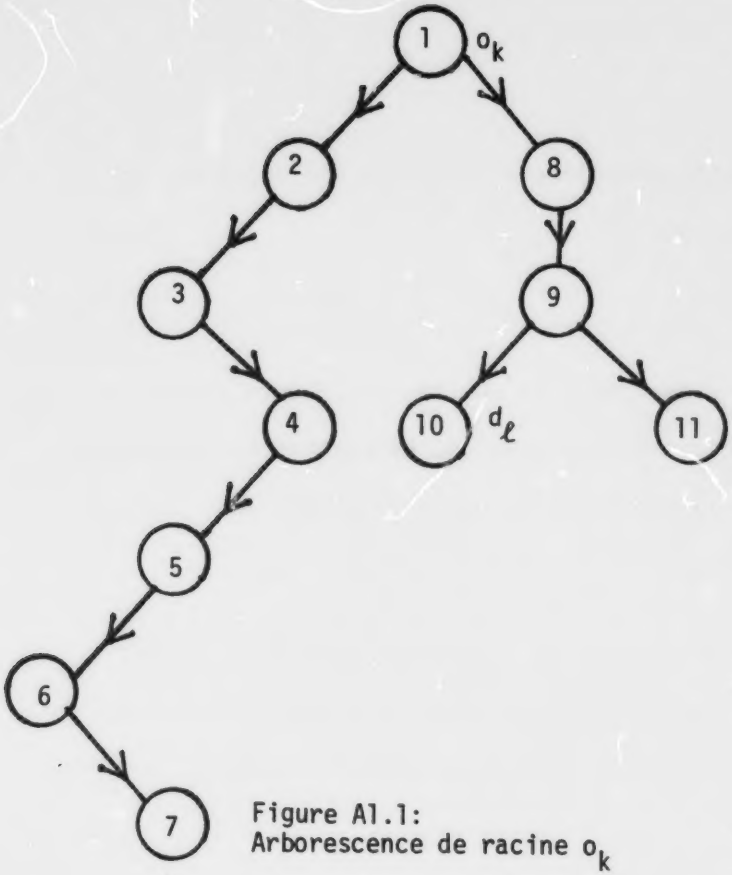


Figure A1.1:
Arborescence de racine o_k

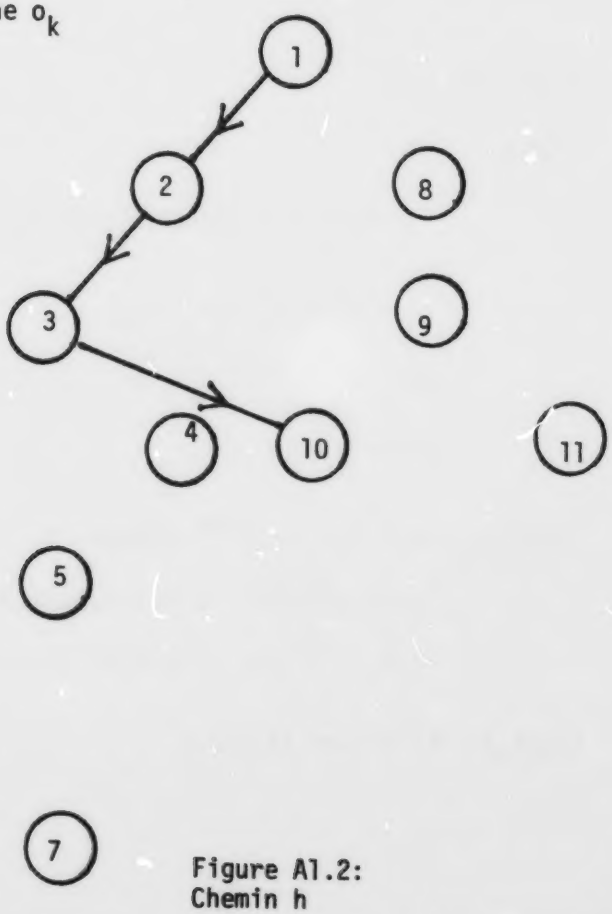


Figure A1.2:
Chemin h

Remarquons finalement que cette procédure n'a pas été implantée sur ordinateur dans le cadre de ce mémoire.

A1.4 Extensions

Une extension de la méthode présentée à la section précédente permet de conserver une suite d'arborescences des plus courts chemins en conservant la première arborescence de la suite et une série de pivots pour chacune des arborescences subséquentes.

Notons par T^0 l'arborescence initiale que l'on doit conserver, et par T^i la $(i+1)^{ième}$ arborescence de la suite. Cette arborescence est conservée sous forme de pivots, lesquels sont déterminés à l'aide de la procédure suivante:

Algorithme P2:

Etape 1: Poser $T_0^i \leftarrow T^i$.

Faire la différence symétrique entre T^0 et T_0^i:

$$T^0 + T_0^i = D_1 \cup D_2$$

où $D_1 \leftarrow T^0 - T_0^i$ et $D_2 \leftarrow T_0^i - T^0$.

Poser $j \leftarrow 1$.

Etape 2: Soit ℓ_j le $j^{ième}$ élément de D_2. Dénotons $\ell_j = (q_1, q_2)$. C'est l'arc entrant. L'arc sortant est le *seul* arc de la forme $(x, q_2) \in T_{j-1}^i$. Dénotons cet arc par $(\tilde{x}, q_2)$.

Etape 3: Effectuer le pivot:

$$T_j^i \leftarrow T_{j-1}^i \cup \{(q_1,q_2)\} - \{(\tilde{x},q_2)\}.$$

Si $j = |D_2|$ terminer.

Poser $j \leftarrow j+1$ et aller en 2.

Le choix de l'arc sortant $(\tilde{x},q_2)$ conserve la propriété d'arborescence (c'est-à-dire que tous les T_j^i, $j=0,\ldots,|D_2|$ sont des arborescences). L'affirmation selon laquelle il n'y a qu'un seul arc de la forme (x,q_2) dans T_{j-1}^i est basée sur une propriété des arborescences, à savoir qu'il y a un et un seul chemin de la racine à tout sommet de l'arborescence.

Considérons, par exemple, les arborescences T^0 et T^1 illustrées par les figures A1.3 et A1.4; on ne fait qu'un seul pivot pour passer de T^0 à T^1 : l'arc entrant est (2,3) et l'arc sortant est (4,3).

Lorsque les différentes arborescences dans la suite sont "assez près" de l'arborescence initiale, cette méthode permet d'économiser de l'espace-mémoire.

Si le nombre moyen d'arc différents entre T^0 et T^i grandit avec i, on peut calculer la suite de pivots qui permet de passer de T^{i-1} à T^i plutôt que celle qui permet de passer de T^0 à T^i ; dans ce cas, cependant, il est beaucoup plus difficile de retracer l'arborescence T^i.

Nous présentons maintenant une autre extension de cette méthode. Glover, Karney et Klingman [8] ont mis au point une structure de données, nommée API ("Augmented Predecessor Index"), destinée à représenter les arbres associés aux bases du simplexe; cette structure de données correspond à

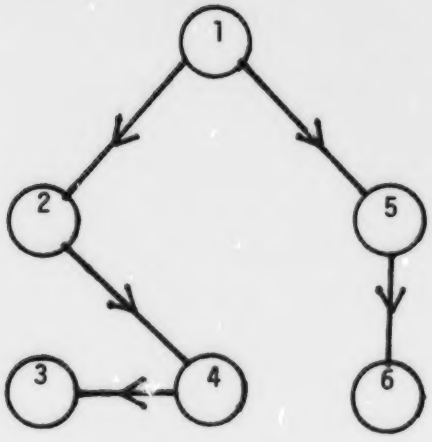


Figure A1.3: T^0

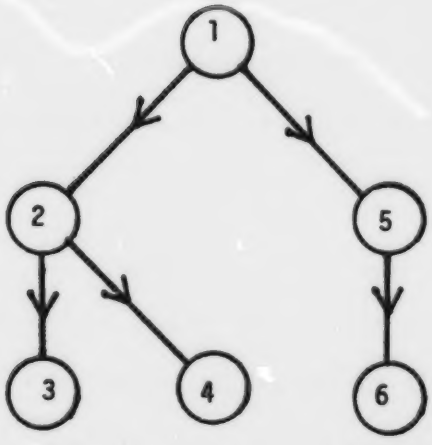


Figure A1.4: T^1

la structure du "préordre" définie par Knuth [9]. Il peut parfois être intéressant, pour certaines applications particulières, de représenter une arborescence selon cette méthode; il faut cependant noter que les pivots, dans ce cas-ci, ne respectent pas la structure d'arborescence (ils respectent cependant la structure d'arbre). L'algorithme P2 devient alors:

Algorithme P3:

Etape 1: Poser $T_0^i \leftarrow T^i$.

Faire $D_1 \leftarrow T^0 - T_0^i$ et $D_2 \leftarrow T_0^i - T^0$.

Poser $j \leftarrow 1$.

Etape 2: Soit ℓ_j le $j^{ième}$ élément de D_2.

2.1: Faire $T_j^i \leftarrow T_{j-1}^i \cup \{\ell_j\}$

T_j^i contient un cycle $C_{i,j}$.

2.2: Trouver un arc Z tel que $Z \in C_{i,j} \cap D_1$.

Faire $D_1 \leftarrow D_1 - \{Z\}$ et $T_j^i \leftarrow T_j^i - \{Z\}$.

(L'arc entrant est ℓ_j et l'arc sortant est Z).

Etape 3: Si $j = |D_2|$ terminer.

Faire $j \leftarrow j + 1$, aller en 2.

L'algorithme précédent est fondé sur les deux propriété suivante d'un arbre:

a) un arbre est un graphe connexe et sans cycle

et

b) lorsqu'on ajoute un arc (i,j) à un arbre A, le graphe résultant comporte un et un seul cycle.

APPENDICE A2

LISTES DES PROGRAMMES

A2.1 Programme LINDYN (algorithme L1, section 4.1)

```
      PROGRAM LINDYN(INPUT,OUTPUT,TAPE1,TAPE2,TAPE3,TAPE4,TAPE5)                20
C                                                                                21
C     LINEARISATION DYNAMIQUE DES FONCTIONS DE DEMANDE                           22
C           VERSION 1                                                            23
C                                                                                24
      COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                           25
      DIMENSION IDEN(10)                                                         40
      INTEGER CURNET,TABIN,ZNEFIL,PTHFIL,SOLFIL                                  50
      DATA CURNET,PTHFIL,TABIN,ZNEFIL,SOLFIL                                     51
     +    /1,2,3,4,5/                                                           60
C                                                                                70
C                                                                                80
      READ 1004,(IDEN(J),J=1,8)                                                  90
      PRINT 1004,(IDEN(J),J=1,8)                                                100
 1004 FORMAT (5X,7A8,A4)                                                        110
C                                                                               120
      CALL TRAFIC(IDEN)                                                         130
C                                                                               140
      END                                                                       150
                                                                                160
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
16377 LINDYN

VARIABLES	SN	TYPE	RELOCATION					
0	CURNET	INTEGER	UNIT	16426	IDEN	INTEGER	ARRAY	
16425	J	INTEGER		1	PTHFIL	INTEGER		UNIT
4	SOLFIL	INTEGER	UNIT	2	TABIN	INTEGER		UNIT
3	ZNEFIL	INTEGER	UNIT					

FILE NAMES	MODE							
0	INPUT	FMT	2043	OUTPUT	FMT	4106	TAPE1	6151 TAPE2
10214	TAPE3		12257	TAPE4		14322	TAPE5	

EXTERNALS	TYPE	ARGS
TRAFIC		1

STATEMENT LABELS
16423 1004 FMT

COMMON BLOCKS	LENGTH
UNIT	5

STATISTICS

PROGRAM LENGTH	538	43
BUFFER LENGTH	163650	7413
CM LABELED COMMON LENGTH	56	5

```
      SUBROUTINE TRAFIC (IDEN)                                          170
C                                                                       180
C                                                                       190
C                                                                       200
      COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                  220
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)   230
      COMMON /OD1/ NCENT, NPAIR, ROW(20), KORIG(20),NOPR(20)            240
      COMMON MEMO(800)                                                  250
      COMMON /RANDF/INDEX(400)                                          260
      COMMON /STAT/ RUNSEC(10)                                          270
      COMMON /PARAM/ EPSI,TIME, MAXIT,PRINT,DEMAND                      280
      DIMENSION IDEN(8), LINK(300)                                      290
      EQUIVALENCE (LINK(1),MEMO(1))                                     310
      EQUIVALENCE (RDSEC,RUNSEC(1)),(SOLSEC,RUNSEC(2)),                 320
     +            (SHSEC,RUNSEC(3)),(PTHSEC,RUNSEC(4))                 330
      INTEGER CURNET,TABIN,ZNEFIL,PTHFIL,SOLFIL                         340
*                                                                       360
      DATA MAXPTH /400/                                                 370
      DATA EPSI,MAXIT,TIME /0.001,200,75.0/                             371
C                                                                       380
 1002 FORMAT(//1H0,20X,"6102I AFFECTATION TERMINEE.....DUREE (SECS)",   390
     +       " MAXIMUM :", F10.0)                                       400
 1003 FORMAT(//1H0,20X,"6103I AFFECTATION TERMINEE ... ITERATION",      410
     +       " MAXIMUM :",I5)                                           420
 1019 FORMAT ("0 SECS CPU UTILISE POUR - ",                             470
     +        "ENTRER LES DONNEES DU RESEAU",T60,"=",F10.2/             480
     +    T24," - ETABLIR LA SOLUTION INITIALE",T60,"=",F10.2/          490
     +    T24," - CALCULER LAMBDA (TOTALE)",T60,"=",F10.2/              500
     +    T24," - TROUVER LES CHEMINS PLUS COURTS",T60,"=",F10.2)       510
 2019 FORMAT (7A8,A4,1X,"IT=",I3,A10)                                   520
 2020 FORMAT(2I4,2F8.2)                                                 530
 2999 FORMAT(1H0,20X,                                                   540
     +          "F O N C T I O N  O B J E C T I V E *,                 550
     +/,3X,*ITER*,8X,*RESEAU*,15X,*DEMANDE*,10X,*(-)TOTALE*,11X,       560
     +*NOMBRE DE CHEMIN*,5X,*SEC. CUMULEES*/)                           570
 3000 FORMAT(3X,I3,3X,3(4X,E16.10),4X,I6,10X,F5.2)                      580
 4000 FORMAT(15F7.4)                                                    620
 4001 FORMAT(8F10.4)                                                    630
C                                                                       640
C  INITIALISATION                                                       650
C                                                                       660
      DO 10 I=1,10                                                      670
   10  RUNSEC(I)=0.0                                                    680
      CALL SECOND (TBEGIN)                                              690
      IPAGE =1                                                          720
      DEMVAL = 0.00                                                     730
C                                                                       740
C--------READ IN NETWORK                                                750
C                                                                       760
      CALL RDNET (CURNET,NCENT)                                         770
C                                                                       780
*------READ TRIP MATRIX OR UPO MATRIX                                   790
*                                                                       800
      CALL RDOD (TABIN,NCENT,ZNEFIL,KORIG)                              810
   60 CALL SECOND(TTOUT)                                                830
      RDSEC=TTOUT-TBEGIN                                                900
      TTIN=TTOUT                                                        910
```

```
      C                                                                 920
      C--------SET UP INITIAL (ALL OR NOTHING) SOLUTION                  930
   60 C                                                                 940
            REWIND ZNEFIL                                               950
            CALL OPENMS (PTHFIL,INDEX,400,0,I)                          961
            DO 40 I=1,NLINK                                             970
            VCUM(I)=0.0                                                 980
   65    40 TRAV(I)=UCOST(I,VCUM(I))                                    990
            NPAIR=0                                                    1000
            DO 42 N=1,NCENT                                            1010
            NOPR(N)=0                                                  1020
            NN=KORIG(N)                                                1030
   70       IF(NN.LE.0)GOTO 42                                         1040
            READ(ZNEFIL,4000) (ROW(J),J=1,NCENT)                       1050
            CALL LOADTR(NN,PTHFIL)                                     1060
            DO 43 I=1,NLINK                                            1070
         43     TRAV(I)=UCOST(I,VCUM(I))                               1080
   75    42 CONTINUE                                                   1100
            NUMCYC = 0                                                 1110
            NBPTH=NPAIR                                                1120
            CALL SECOND(TTOUT)                                         1130
            SOLSEC=TTOUT-TTIN                                          1140
   80 C                                                                1150
            TTIN=TTOUT                                                 1160
      C                                                                1170
      C--------PRINT HEADING FOR ITERATION LOG                         1180
        100 IPAGE = IPAGE + 1                                          1300
   85       PRINT 2999                                                 1310
      C                                                                1330
            KOPT=0                                                     1340
        200 DO 205 J=1,NPAIR                                           1350
            CALL MONEQU (J,PTHFIL,KOPT,EPSI)                           1360
   90       IF(KOPT.EQ.NPAIR)GOTO 250                                  1370
        205 CONTINUE                                                   1380
        250 CONTINUE                                                   1390
      C                                                                1400
            CALL OBFUN(RESVAL,DEMVAL,SYSVAL)                           1470
   95       CALL SECOND(TTOUT)                                         1480
            TTOUT=TTOUT-TBEGIN                                         1490
            NUMCYC=NUMCYC+1                                            1510
            TOTVAL = DEMVAL - RESVAL                                   1520
            PRINT 3000,NUMCYC,RESVAL,DEMVAL,TOTVAL,NBPTH,TTOUT        1521
  100       IF(KOPT.EQ.NPAIR)GOTO 300                                  1522
      C CHECK THE NUMBER OF ALLOWABLE CYCLES                           1531
        210 IF(NUMCYC.LT.MAXIT)GOTO 220                                1540
            PRINT 1003,MAXIT                                           1550
            GOTO 500                                                   1560
  105 C                                                                1570
      C CHECK THE RUNNING TIME                                         1580
        220 IF(TTOUT.LT.TIME)GOTO 200                                  1590
            PRINT 1002,TIME                                            1600
            GOTO 500                                                   1610
  110   300 NADD=0                                                     1620
            NDEL=0                                                     1630
            KOPT=0                                                     1640
            K2=0                                                       1650
            DO 301 N=1,NCENT                                           1660
                                                                       1670
```

```
115                NN=KORIG(N)                                                    1680
                   IF(NN.LE.0)GOTO 301                                            1690
                C                                                                 1700
                   K1=K2+1                                                        1710
                   K2=NOPR(N)+K2                                                  1720
120             C                                                                 1730
                   CALL REVPTH(NN,K1,K2,PTHFIL,NADD,NDEL,EPSI)                    1740
                 301 CONTINUE                                                     1750
                   NBPTH=NBPTH + NADD - NDEL                                      1760
                   CALL OUTPTH(NPAIR,PTHFIL)                                      1761
125                IF(NADD.GT.0)GOTO 200                                          1770
                C                                                                 1780
                C--------OPTIMUM IS REACHED OR TIME LIMIT OR CYCLE LIMIT          1790
                C                                                                 1800
                 500 CONTINUE                                                     1810
120                IPAGE =IPAGE + 1                                               1820
                   PRINT 1019,(RUNSEC(I),I=1,4)                                   1840
                *                                                                 1842
                C                                                                 1850
                C---------SAVE SOLUTION VALUES                                   1900
135                REWIND SOLFIL                                                  1910
                   CALL DATE (TDAT)                                               1920
                   WRITE (SOLFIL,2019) IDEN,NUMCYC,TDAT                           1930
                   DO 90 NO=1,NNODE                                               1940
                   J=NOLAB(NO)                                                    1950
140                IF(J.LE.0)GOTO 90                                              1960
                   II=NO                                                          1970
                  91 II=II+1                                                      1980
                   JJ=NOLAB(II)-1                                                 1990
                   IF(JJ.LE.0)GOTO 91                                             2000
145                DO 92 N=J,JJ                                                   2010
                C                                                                 2020
                   PRINT 2020, NO,ND(N),VCUM(N),TRAV(N)                           2030
                  92 CONTINUE                                                     2040
                C                                                                 2050
150                90 CONTINUE                                                    2060
                C                                                                 2070
                   REWIND ZNEFIL                                                  2080
                   N=1                                                            2090
                   K2=0                                                           2100
155                DO 99 N=1,NCENT                                                2110
                   DO 93 I=1,NCENT                                                2120
                  93 ROW(I)=0                                                     2130
                   IF(KORIG(N).LE.0)GOTO 96                                       2150
                   K1=K2 + 1                                                      2160
160                K2 = NOPR(N) + K2                                              2170
                   DO 94 J=K1,K2                                                  2180
                   CALL READMS(PTHFIL,LINK,300,J)                                 2190
                   ROW(LINK(4))= LINK(M)/10000.0                                  2200
                  94 CONTINUE                                                     2220
165                96  PRINT 4001, (ROW(I),I=1,NCENT)                            2230
                  99 CONTINUE                                                     2250
                *                                                                 2260
                   ENDFILE SOLFIL                                                 2270
                   ENDFILE ZNEFIL                                                 2280
170             *                                                                 2300
                   RETURN                                                         2310
```

```
      SUBROUTINE OBFJN(RESVAL,DEMVAL,SYSVAL)                                  2330
*                                                                             2340
      COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                        2350
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)         2360
      COMMON /OD1/ NCENT, NPAIR, ROW(20), KORIG(20),NOPR(20)                  2370
      COMMON MEMO1(800)                                                       2371
      DIMENSION LINK(300)                                                     2372
      INTEGER PTHFIL                                                          2373
      EQUIVALENCE (LINK(1),MEMO1(1))                                          2381
*                                                                             2390
      RESVAL= DEMVAL= SYSVAL = 0.0                                            2400
      DO 10 I=1,NLINK                                                         2410
      RESVAL=RESVAL + TCOST(I,VCUM(I))                                        2420
   10 SYSVAL=SYSVAL + VCUM(I)*TRAV(I)                                         2430
      DO 40 J=1,NPAIR                                                         2440
      CALL READMS(PTHFIL,LINK,300,J)                                          2510
      PENTE=LINK(299)/10000.                                                  2520
      ORD=LINK(300)/10000.                                                    2521
      UPQ=LINK(1)/10000.                                                      2522
      THETA=LINK(2)/10000.                                                    2523
      GPQ=FCTDEM(UPQ,THETA)                                                   2524
      DEMVAL=DEMVAL+(PENTE*GPQ*GPQ/2.0)+(ORD*GPQ)                             2534
   40 CONTINUE                                                                2540
*                                                                             2550
      RETURN                                                                  2560
      END                                                                     2570
                                                                              2580
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS

4	OBFJN

VARIABLES

SN		TYPE		RELOCATION
0	CURNET	REAL		UNIT
122	GPQ	REAL		
115	J	INTEGER		
0	LINK	INTEGER	ARRAY	/ /
0	NCENT	INTEGER		OD1
1	NLINK	INTEGER		NET1
2	NOLAB	INTEGER	ARRAY	NET1
1	NPAIR	INTEGER		OD1
116	PENTE	REAL		
0	RESVAL	REAL		F.P.
4	SOLFIL	REAL		UNIT
2	TABIN	REAL		UNIT
1214	TRAV	REAL	ARRAY	NET1
2034	VCUM	REAL	ARRAY	NET1
0	DEMVAL	REAL		F.P.
114	I	INTEGER		
26	KORIG	INTEGER	ARRAY	OD1
0	MEMO1	INTEGER	ARRAY	/ /
374	ND	INTEGER	ARRAY	NET1
0	NNODE	INTEGER		NET1
52	NOPR	INTEGER	ARRAY	OD1
117	ORD	REAL		
1	PTHFIL	INTEGER		UNIT
2	ROW	REAL	ARRAY	OD1
0	SYSVAL	REAL		F.P.
121	THETA	REAL		
120	UPQ	REAL		
3	ZNEFIL	REAL		UNIT

EXTERNALS

	TYPE	ARGS		
FCTDEM	REAL	2		
TCOST	REAL	2	READMS	4

```
      SUBROUTINE EQUIL(EPSI,ELAST,U0I,G0I,U1I,UB,                       3860
     +PENTE,ORD)                                                        3861
*                                                                       3870
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)   3880
      COMMON MEMO1(800),MEMO2(550)                                      3890
      DIMENSION LINK(300),PATH(50,5),IFLOW(5),LPJ(5)                    3900
      DIMENSION DELTA(100)                                              3910
      DIMENSION XTRAV(400),XVCUM(400)                                   3911
      INTEGER PATH                                                      3920
      EQUIVALENCE (DELTA(1),MEMO1(301))                                 3930
      EQUIVALENCE (LINK(1),MEMO1(1)),(PATH(1,1),LINK(16)),              3940
     + (NO,LINK(3)),(NDEST,LINK(4)),(NP,LINK(5)),                       3950
     +       (IFLOW(1),LINK(5)),(LPJ(1),LINK(11))                       3960
      DIMENSION  DRT(6),X(6),U(6)                                       3970
      LOGICAL ELAST                                                     3980
      REAL LAMBDA                                                       3990
      DATA GMAX/1000.0/                                                 4000
      DATA ACCUR/1.0E-4/                                                4001
      DATA XLIM/1.0/                                                    4002
*                                                                       4010
*                                                                       4020
      IF(ELAST) GOTO 11                                                 4021
      DO 10 I=1,NLINK                                                   4022
      XTRAV(I)=TRAV(I)                                                  4023
      XVCUM(I)=VCUM(I)                                                  4024
   10 CONTINUE                                                          4025
   11 LAMBDA=0.0                                                        4026
*                                                                       4050
      CALL SECOND(TTIN)                                                 4070
      IF(NP.EQ.0)RETURN                                                 4090
      NPA=NP                                                            4100
      J1=NP + 1                                                         4110
      U(J1)=0                                                           4120
      X(J1)=G0I                                                         4130
      DO 20 I=1,NP                                                      4140
   20 X(I)= IFLOW(I)/10000.0                                            4150
      IF(.N.ELAST)GOTO 100                                              4160
      NPA=J1                                                            4170
      U(J1)=U0I                                                         4180
*                                                                       4190
*                                                                       4200
  100 DO 120 I=1,NP                                                     4210
      N=LPJ(I)                                                          4220
      SUM=0                                                             4230
      DO 121 J=1,N                                                      4240
  121 SUM=SUM + TRAV(PATH(J,I))                                         4250
  120 U(I)=SUM                                                          4260
*                                                                       4270
*                                                                       4280
      DO 140 I=1,J1                                                     4290
  140 DRT(I)=0                                                          4300
*                                                                       4310
*                                                                       4320
      UBA=1.0E9                                                         4330
      DO 150I= 1.NPA                                                    4340
      IF(UBA.LE.U(I))GOTO 150                                           4350
      UBA=U(I)                                                          4360
```

```
      IBA=I
  150 CONTINUE
*
      SUM=0
      ALPHA=0
      DO 160 J=1,NP
      D=U(J)-UBA
      GAP=2.0*D/(UBA+U(J)+0.001)
      IF(GAP.LE.EPSI.OR. X(J).LE.0.1)GOTO 160
      DRT(J)=-D*X(J)
      SUM=SUM + DRT(J)
      IF(ALPHA.LT.D)ALPHA=D
  160 CONTINUE
*
      IF(J1.EQ.IBA)GOTO 170
      SUM= - SUM
      IF(.N.ELAST)GOTO 170
      GAP=(U(J1)-UBA)*2.0/(U(J1)+UBA)
      IF(GAP.LE.EPSI)GOTO 170
      DRT(J1)=(U(J1)-UBA)*GMAX
      SUM= DRT(J1) +SUM
  170 DRT(IBA)= SUM
*
      IF(ABS(SUM).LE.0.0)GOTO 500
      IF(ALPHA.LE.0.0)ALPHA=1.0
  190 DO 200 J=1,NPA
  200 DRT(J)=DRT(J)/ALPHA
*
      DO 205 I=1,NLINK
  205 DELTA(I)=0.0
*
      DO 210 I=1,NP
      N=LPJ(I)
      D=DRT(I)
      IF(ABS(D).LE.0.0)GOTO 210
      DO 215 J=1,N
  215 DELTA(PATH(J,I))= DELTA(PATH(J,I)) +D
  210 CONTINUE
*
      XX=X(J1)
      DD=DRT(J1)
      CALL GSM(LAMBDA,XX,DD,PENTE,ORD,ELAST,XLIM,ACCUR)
*
  281 DO 300 I=1.NLINK
      IF(ABS(DELTA(I)).LE.0.0)GOTO 300
      VCUM(I)= VCUM(I) + LAMBDA*DELTA(I)
      IF(VCUM(I).LE.0.1)VCUM(I)=0
      TRAV(I)=UCOST(I,VCUM(I))
  300 CONTINUE
      DO 310 I=1,NPA
  310 X(I)= X(I) + LAMBDA*DRT(I)
      IF(.NOT.ELAST) GOTO 100
      U(J1)=PENTE*X(J1)+ORD
      GOTO 100
*
*
*
```

```
                500 IF(ELAST) GOTO 501                                          4901
                    DO 502 I=1.NLINK                                            4904
                    VCUM(I)=XVCUM(I)                                            4905
                    TRAV(I)=XTRAV(I)                                            4906
                502 CONTINUE                                                    4907
                    UII=UBA                                                     4908
                    RETURN                                                      4909
                501 DO 510 I=1,NP                                               4910
                    IFLOW(I)=X(I)*10000                                         4911
                510 CONTINUE                                                    4912
                    U(J1)=PENTE*X(J1)+ORD                                       4913
                    UB=U(J1)                                                    4914
                    LINK(1)=U(J1)*10000                                         4917
                    RETURN                                                      4918
                    END                                                         4923
```

CARD NR. SEVERITY DETAILS DIAGNOSIS OF PROBLEM

94 I NP THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.
123 I NP THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 EQUIL

VARIABLES	SN	TYPE	RELOCATION						
470	ACCUR	REAL			513	ALPHA	REAL		
514	D	REAL			517	DD	REAL		
454	DELTA	REAL	ARRAY	/ /	2160	DRT	REAL	ARRAY	
0	ELAST	LOGICAL		F.P.	0	EPSI	REAL		F.P.
515	GAP	REAL			467	GMAX	REAL		
0	GOI	REAL		F.P.	502	I	INTEGER		
512	IBA	INTEGER			5	IFLOW	INTEGER	ARRAY	/ /
510	J	INTEGER			505	J1	INTEGER		
501	LAMBDA	REAL			0	LINK	INTEGER	ARRAY	/ /
12	LPJ	INTEGER	ARRAY	/ /	0	MEMO1	INTEGER	ARRAY	/ /
1440	MEMO2	INTEGER	ARRAY	/ /	506	N	INTEGER		
374	ND	INTEGER	ARRAY	NET1	3	NDEST	INTEGER		/ /
1	NLINK	INTEGER		NET1	0	NNODE	INTEGER		NET1
2	NO	INTEGER		/ /	2	NOLAB	INTEGER	ARRAY	NET1
4	NP	INTEGER		/ /	504	NPA	INTEGER		
0	ORD	REAL		F.P.	17	PATH	INTEGER	ARRAY	/ /
0	PENTE	REAL		F.P.	507	SUM	REAL		
1214	TRAV	REAL	ARRAY	NET1	503	TTIN	REAL		
2174	U	REAL	ARRAY		0	UB	REAL		F.P.
511	UBA	REAL			0	UOI	REAL		F.P.
0	UII	REAL		F.P.	2034	VCUM	REAL	ARRAY	NET1
2166	X	REAL	ARRAY		471	XLIM	REAL		
520	XTRAV	REAL	ARRAY		1340	XVCUM	REAL	ARRAY	

```
      1                 SUBROUTINE GSM(LAMBDA,GP,DG,PENTE,ORD,ELAST,XLIM,ACCUR)                4924
                        COMMON /STAT/RUNSEC(10)                                                4925
                        LOGICAL ELAST                                                          4926
                        EQUIVALENCE(SHSEC,RUNSEC(3))                                           4927
      5                 REAL LAMBDA                                                            4928
                    *                                                                          4929
                        CALL SECOND(TTIN)                                                      4930
                        TAU=(1.0+SQRT(5.))*.5                                                  4931
                    *                                                                          4932
     10                 G=0.0                                                                  4933
                        D=XLIM                                                                 4934
                        XB=D/TAU                                                               4935
                        XA=D-XB                                                                4936
                    *                                                                          4937
     15                 FXA=FCTVLU(XA,GP,DG,PENTE,ORD,ELAST)                                   4938
                        FXB=FCTVLU(XB,GP,DG,PENTE,ORD,ELAST)                                   4939
                    *                                                                          4940
                   77   IF(FXA-FXB) 80,80,78                                                   4941
                   78   G=XA                                                                   4942
     20                 XA=XB                                                                  4943
                        XB=G+D-XA                                                              4944
                        FXA=FXB                                                                4945
                        FXB=FCTVLU(XB,GP,DG,PENTE,ORD,ELAST)                                   4946
                        GOTO 82                                                                4947
     25             *                                                                          4948
                   80   D=XB                                                                   4949
                        XB=XA                                                                  4950
                        XA=G+D-XB                                                              4951
                        FXB=FXA                                                                4952
     30                 FXA=FCTVLU(XA,GP,DG,PENTE,ORD,ELAST)                                   4953
                    *                                                                          4954
                   82   IF(D-G-ACCUR) 83,83,77                                                 4955
                    *                                                                          4956
                   83   LAMBDA=XB                                                              4957
     35                 VAL=FXB                                                                4958
                        IF(FXB.LT.FXA) GOTO 1000                                               4959
                        VAL=FXA                                                                4960
                        LAMBDA=XA                                                              4961
                    *                                                                          4962
     40            1000 CALL SECOND (TTOUT)                                                    4963
                        SHSEC=SHSEC+TTOUT-TTIN                                                 4964
                        RETURN                                                                 4965
                        END                                                                    4966
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS

4	GSM

VARIABLES	SN	TYPE	RELOCATION				
0	ACCUR	REAL	F.P.	213	D	REAL	
0	DG	REAL	F.P.	0	ELAST	LOGICAL	F.P.
216	FXA	REAL		217	FXB	REAL	

```
      1              REAL FUNCTION FCTVLU(LAMBDA,G,DG,PENTE,ORD,ELAST)                    4967
                     COMMON /NET1/NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)      4968
                     COMMON MEMO1(800),MEMO2(550)                                         4969
                     REAL LAMBDA                                                          4970
      5              DIMENSION DELTA(100)                                                 4971
                     LOGICAL ELAST                                                        4972
                     EQUIVALENCE(DELTA(1),MEMO1(301))                                     4973
                *                                                                         4974
                     F=0.0                                                                4975
     10              DO 10 I=1,NLINK                                                      4976
                     VOL=VCUM(I)+(LAMBDA*DELTA(I))                                        4977
                  10 F=F+TCOST(I,VOL)                                                     4978
                     FCTVLU=F                                                             4979
                     IF(.NOT.ELAST) RETURN                                                4980
     15              GG=G+LAMBDA*DG                                                       4981
                     F=F-(PENTE*GG*GG/2.0)-(ORD*GG)                                       4982
                     FCTVLU=F                                                             4983
                     RETURN                                                               4984
                     END                                                                  4985
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
5 FCTVLU

VARIABLES	SN	TYPE		RELOCATION					
454	DELTA	REAL	ARRAY	/ /	0	DG	REAL		F.P.
0	ELAST	LOGICAL		F.P.	65	F	REAL		
64	FCTVLU	REAL			0	G	REAL		F.P.
70	GG	REAL			66	I	INTEGER		
0	LAMBDA	REAL		F.P.	0	MEMO1	INTEGER	ARRAY	/ /
1440	MEMO2	INTEGER	ARRAY	/ /	374	ND	INTEGER	ARRAY	NET1
1	NLINK	INTEGER		NET1	0	NNODE	INTEGER		NET1
2	NOLAB	INTEGER	ARRAY	NET1	0	ORD	REAL		F.P.
0	PENTE	REAL		F.P.	1214	TRAV	REAL	ARRAY	NET1
2034	VCUM	REAL	ARRAY	NET1	67	VOL	REAL		

EXTERNALS	TYPE	ARGS
TCOST	REAL	2

STATEMENT LABELS
0 10

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
23	10	* I	10 12	128	EXT REFS

COMMON BLOCKS	LENGTH
NET1	1452
/ /	1350

STATISTICS

PROGRAM LENGTH	1058	69
CM LABELED COMMON LENGTH	26548	1452

```
      SUBROUTINE MONEQU(IPAIR,PTHFIL,KOPT,EPSI)                          8030
      COMMON MEMO1(800),MEMO2(550)                                       8040
      INTEGER PTHFIL                                                     8050
      LOGICAL FLAG,ELAST                                                 8051
      DIMENSION LINK(300)                                                8060
      EQUIVALENCE(LINK(1),MEMO1(1))                                      8070
*                                                                        8080
      CALL READMS(PTHFIL,LINK,300,IPAIR)                                 8081
*                                                                        8090
      PENTE=ORD=0.0                                                      8094
      KOPT=KOPT+1                                                        8100
      FLAG=.TRUE.                                                        8110
      U0I=LINK(1)/10000.                                                 8120
      THETA=LINK(2)/10000.                                               8130
*                                                                        8140
   10 G0I=FCTDEM(U0I,THETA)                                              8150
      ELAST=.FALSE.                                                      8160
*                                                                        8170
      CALL EQUIL(EPSI,ELAST,U0I,G0I,U1I,UB,PENTE,ORD)                    8180
*                                                                        8190
      G1I=FCTDEM(U1I,THETA)                                              8200
      DIFF=2.0*ABS(U1I-U0I)/(U1I+U0I+0.001)                              8201
      IF(DIFF.LE.EPSI) GOTO 100                                          8202
      FLAG=.FALSE.                                                       8220
*                                                                        8230
      PENTE=(U1I-U0I)/(G1I-G0I)                                          8240
      ORD=-1.0*G0I*PENTE+U0I                                             8250
      ELAST=.TRUE.                                                       8260
*                                                                        8280
      CALL EQUIL(EPSI,ELAST,U0I,G0I,U1I,UB,PENTE,ORD)                    8290
*                                                                        8300
      U0I=UB                                                             8310
      GOTO 10                                                            8320
*                                                                        8330
  100 IF(.NOT.FLAG) KOPT=1                                               8340
      LINK(299)=PENTE*10000                                              8350
      LINK(300)=ORD*10000                                                8351
      CALL WRITMS(PTHFIL,LINK,300,IPAIR,1,1)                             8352
      RETURN                                                             8353
      END                                                                8360
                                                                         8370
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 MONEQU

VARIABLES	SN	TYPE		RELOCATION					
174	DIFF	REAL			163	ELAST	LOGICAL		
0	EPSI	REAL		F.P.	162	FLAG	LOGICAL		
170	G0I	REAL			173	G1I	REAL		
0	IPAIR	INTEGER		F.P.	0	KOPT	INTEGER		F.P.
0	LINK	INTEGER	ARRAY	/ /	0	MEMO1	INTEGER	ARRAY	/ /
1440	MEMO2	INTEGER	ARRAY	/ /	165	ORD	REAL		

A2.2. Programme LINDYN2 (algorithme L2, section 4.2)

```
      1              PROGRAM LINDY2(INPUT,OUTPUT,TAPE1,TAPE2,TAPE3,TAPE4,TAPE5)                    20
                     COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                              40
                     DIMENSION IDEN(10)                                                            50
                     INTEGER CURNET,TABIN,ZNEFIL,PTHFIL,SOLFIL
      5              DATA CURNET,PTHFIL,TABIN,ZNEFIL,SOLFIL                                        51
                    +    /1,2,3,4,5/                                                             60
                C                                                                                 70
                C                                                                                 80
                     READ 1004,(IDEN(J),J=1,8)                                                    90
     10              PRINT 1004,(IDEN(J),J=1,8)                                                   100
                1004 FORMAT (5X,7A8,A4)                                                           110
                C                                                                                 120
                     CALL TRAFIC(IDEN)                                                            130
                C                                                                                 140
     15              END                                                                          150
                                                                                                  160
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
16377 LINDY2

VARIABLES	SN	TYPE	RELOCATION						
0	CURNET	INTEGER	UNIT	16426	IDEN	INTEGER	ARRAY		
16425	J	INTEGER		1	PTHFIL	INTEGER		UNIT	
4	SOLFIL	INTEGER	UNIT	2	TABIN	INTEGER		UNIT	
3	ZNEFIL	INTEGER	UNIT						

FILE NAMES								
0	INPUT	FMT	2043	OUTPUT	FMT	4106	TAPE1	6151	TAPE2
10214	TAPE3		12257	TAPE4		14322	TAPE5		

EXTERNALS	TYPE	ARGS
TRAFIC		1

STATEMENT LABELS
16423 1004 FMT

COMMON BLOCKS	LENGTH
UNIT	5

STATISTICS

PROGRAM LENGTH	538	43
BUFFER LENGTH	163658	7413
CM LABELED COMMON LENGTH	58	5

```
      SUBROUTINE MONEQU(IPAIR,PTHFIL,KOPT,EPSI)                          8030
      COMMON MEMO1(800),MEMO2(550)                                       8040
      INTEGER PTHFIL                                                     8050
      LOGICAL FLAG,ELAST                                                 8051
      DIMENSION LINK(300)                                                8060
      EQUIVALENCE(LINK(1),MEMO1(1))                                      8070
      DATA EPS2/0.1/                                                     8071
*                                                                        8080
      CALL READMS(PTHFIL,LINK,300,IPAIR)                                 8081
*                                                                        8090
 9999 FORMAT(* MONEQUIL, IPAIR  = *,I5)                                  8093
      PENTE=ORD=0.0                                                      8094
      KOPT=KOPT+1                                                        8100
      FLAG=.TRUE.                                                        8110
      U0I=LINK(1)/10000.                                                 8120
      THETA=LINK(2)/10000.                                               8130
*                                                                        8140
   10 G0I=FCTDEM(U0I,THETA)                                              8150
      ELAST=.FALSE.                                                      8160
*                                                                        8170
      CALL EQUIL(EPSI,ELAST,U0I,G0I,U1I,UB,GB,PENTE,ORD)                 8180
*                                                                        8190
      G1I=FCTDEM(U1I,THETA)                                              8200
      DIFF=2.0*ABS(U1I-U0I)/(U1I+U0I+EPSI)                               8201
      IF(DIFF.LE.EPSI) GOTO 100                                          8202
      FLAG=.FALSE.                                                       8220
*                                                                        8230
      PENTE=(U1I-U0I)/(G1I-G0I)                                          8240
      ORD=-1.0*G0I*PENTE+U0I                                             8250
      ELAST=.TRUE.                                                       8280
*                                                                        8290
      CALL EQUIL(EPSI,ELAST,U0I,G0I,U1I,UB,GB,PENTE,ORD)                 8300
      GPB=FCTDEM(UB,THETA)                                               8301
      PENTE1=(G1I-GPB)/(U1I-UB)                                          8302
      ORD1=G1I-PENTE1*U1I                                                8303
      PENTE2=(GB-G0I)/(UB-U1I)                                           8304
      ORD2=GB-PENTE2*UB                                                  8305
*                                                                        8306
      U0I=(ORD2-ORD1)/(PENTE1-PENTE2)                                    8307
      IF(U0I.EQ.UB) CALL EXIT                                            8308
*                                                                        8310
      GOTO 10                                                            8330
*                                                                        8340
  100 IF(.NOT.FLAG) KOPT=1                                               8350
      LINK(299)=PENTE*10000                                              8351
      LINK(300)=ORD*10000                                                8352
      CALL WRITMS(PTHFIL,LINK,300,IPAIR,1,1)                             8353
      RETURN                                                             8360
      END                                                                8370
```

SYMBOLIC REFERENCE MAP (R=1)

A2.3. Programme GRG (algorithme GRG1, section 5.2)

```
      PROGRAM GRG        73/74   OPT=0 TRACE                    FTN 4.6+428        27/04/78  10.30.48       PAGE     1

     1                PROGRAM GRG(INPUT,OUTPUT,TAPE1,TAPE2,TAPE3,TAPE4,TAPE5)                              130
                     *                                                                                    131
                     * METHODE DU GRADIENT REDUIT GENERALISE --- AFFECTATION                              132
                     *     DU TRAFIC AVEC DEMANDE ELASTIQUE SANS INVERSE                                  133
     5               *                                                                                    134
                     *                                                                                    135
                          COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                                140
                          DIMENSION IDEN(10)                                                              150
                          INTEGER CURNET,TABIN,ZNEFIL,PTHFIL,SOLFIL                                       160
    10                    DATA CURNET,PTHFIL,TABIN,ZNEFIL,SOLFIL                                          170
                         +    /1,2,3,4,5/                                                                180
                     *                                                                                    190
                     *                                                                                    200
                          READ 1004,(IDEN(J),J=1,8)                                                       210
    15                    PRINT 1004,(IDEN(J),J=1,8)                                                      220
                      1004 FORMAT (5X,7A8,A4)                                                              230
                     *                                                                                    240
                          CALL TRAFIC(IDEN)                                                               250
                     *                                                                                    260
    20                    END                                                                             270


      SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
16377  GRG

VARIABLES     SN  TYPE            RELOCATION
    0  CURNET     INTEGER             UNIT          16426  IDEN      INTEGER    ARRAY
16425  J          INTEGER                             1  PTHFIL    INTEGER             UNIT
    4  SOLFIL     INTEGER             UNIT            2  TABIN     INTEGER             UNIT
    3  ZNEFIL     INTEGER             UNIT

FILE NAMES       MODE
    0  INPUT      FMT          2043  OUTPUT     FMT          4106  TAPE1                6151  TAPE2
10214  TAPE3                  12257  TAPE4                  14322  TAPE5

EXTERNALS          TYPE   ARGS
      TRAFIC                   1

STATEMENT LABELS
16423  1004     FMT

COMMON BLOCKS   LENGTH
      UNIT          5

STATISTICS
  PROGRAM LENGTH                 538      43
  BUFFER LENGTH                163658    7413
  CM LABELED COMMON LENGTH         58       5
```

```
      SUBROUTINE TRAFIC (IDEN)                                          280
C                                                                       290
C                                                                       300
C                                                                       310
      COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                  320
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)   330
      COMMON /OD1/ NCENT, NPAIR, ROW(30), KORIG(30),NOPR(30)            340
      COMMON MEMO1(900),MEMO2(550),SUBA(10)                             350
      COMMON /RANDF/INDEX(800)                                          360
      COMMON /STAT/ RUNSEC(10)                                          370
      COMMON /PARAM/ EPSI,TIME, MAXIT,PRINT,DEMAND                      380
      DIMENSION IDEN(8), LINK(300)                                      390
      EQUIVALENCE (LINK(1),MEMO1(1))                                    410
      EQUIVALENCE (RDSEC,RUNSEC(1)),(SOLSEC,RUNSEC(2)),                 420
     +            (SHSEC,RUNSEC(3)),(PTHSEC,RUNSEC(4))                 430
      INTEGER CURNET,TABIN,ZNEFIL,PTHFIL,SOLFIL                         440
*                                                                       460
      DATA MAXPTH /400/                                                 470
      DATA EPSI,MAXIT,TIME /0.01,200,75.0/                              480
C                                                                       490
 1002 FORMAT(///1H0,20X,"6102I AFFECTATION TERMINEE.....DUREE (SECS)",  500
     +       " MAXIMUM :", F10.0)                                       510
 1003 FORMAT(///1H0,20X,"6103I AFFECTATION TERMINEE ... ITERATION",     520
     +       " MAXIMUM :",I8)                                           530
 1019 FORMAT ("0 SECS CPU UTILISE POUR - ",                             540
     +        "ENTRER LES DONNEES DU RESEAU",T60,"=",F10.2/             550
     +    T24," - ETABLIR LA SOLUTION INITIALE",T60,"=",F10.2/          560
     +    T24," - CALCULER LAMBDA (TOTALE)",T60,"=",F10.2/              570
     +    T24," - TROUVER LES CHEMINS PLUS COURTS",T60,"=",F10.2)       580
 2019 FORMAT (7A8,A4,1X,"IT=",I3,A10)                                   590
 2020 FORMAT(2I4,2F8.2)                                                 600
 2999 FORMAT(1H0,20X,                                                   610
     +          "F O N C T I O N  O B J E C T I V E ",                  620
     +/,3X,"ITER",8X,"RESEAU",15X,"DEMANDE",10X,"  TOTALE",11X,        630
     +"NOMBRE DE CHEMIN",5X,"SEC. CUMULEES"/)                           640
 3000 FORMAT(3X,I3,3X,3(4X,E16.10),4X,I6,10X,F5.2)                      650
 4000 FORMAT(15F7.4)                                                    660
 4001 FORMAT(8F10.4)                                                    670
C                                                                       680
C  INITIALISATION                                                       690
C                                                                       700
      DO 10 I=1,10                                                      710
   10  RUNSEC(I)=0.0                                                    720
      CALL SECOND (TBEGIN)                                              730
      IPAGE =1                                                          750
      DEMVAL = 0.00                                                     760
C                                                                       780
C--------READ IN NETWORK                                                790
C                                                                       800
      CALL RDNET (CURNET,NCENT)                                         810
C                                                                       820
*------READ TRIP MATRX OR UPC MATRIX                                    830
*                                                                       840
      CALL RDOD (TABIN,NCENT,ZNEFIL,KORIG)                              850
   60 CALL SECOND(TTOUT)                                                860
      RDSEC=TTOUT-TBEGIN                                                870
      TTIN=TTOUT                                                        880
```

```
      C                                                                 890
      C--------SET UP INITIAL (ALL OR NOTHING) SOLUTION                  900
60    C                                                                 910
            REWIND ZNEFIL                                               920
            CALL OPENMS (PTHFIL,INDEX,800,0,I)                          930
            DO 40 I=1,NLINK                                             940
            VCUM(I)=0.0                                                 950
65       40 TRAV(I)=UCOST(I,VCUM(I))                                    960
            NPAIR=0                                                     970
            DO 42 N=1,NCENT                                             980
            NOPR(N)=0                                                   990
            NN=KORIG(N)                                                1000
70          IF(NN.LE.0)GOTO 42                                         1010
            READ(ZNEFIL,4000) (ROW(J),J=1,NCENT)                       1020
            CALL LOADTR(NN,PTHFIL)                                     1030
            DO 43 I=1,NLINK                                            1040
         43     TRAV(I)=UCOST(I,VCUM(I))                               1050
75       42 CONTINUE                                                   1060
            NUMCYC = 0                                                 1070
            NBPTH=NPAIR                                                1080
            CALL SECOND(TTOUT)                                         1090
            SOLSEC=TTOUT-TTIN                                          1100
80    C                                                                1110
            TTIN=TTOUT                                                 1120
      C                                                                1130
      C--------PRINT HEADING FOR ITERATION LOG                         1140
        100 IPAGE = IPAGE + 1                                          1150
85          PRINT 2999                                                 1160
      C                                                                1170
            KOPT=0                                                     1180
        200 DO 205 J=1,NPAIR                                           1190
            CALL EQUIL (J,PTHFIL,KOPT,EPSI)                            1200
90          IF(KOPT.EQ.NPAIR)GOTO 250                                  1210
        205 CONTINUE                                                   1220
        250 CONTINUE                                                   1230
      C                                                                1240
            CALL OBFJN(RESVAL,DEMVAL,SYSVAL)                           1250
95          CALL SECOND(TTOUT)                                         1260
            TTOUT=TTOUT-TBEGIN                                         1270
            NUMCYC=NUMCYC+1                                            1280
            TOTVAL=RESVAL+DEMVAL                                       1290
            PRINT 3000,NUMCYC,RESVAL,DEMVAL,TOTVAL,NBPTH,TTOUT         1300
100         IF(KOPT.EQ.NPAIR)GOTO 300                                  1310
      C CHECK THE NUMBER OF ALLOWABLE CYCLES                           1320
        210 IF(NUMCYC.LT.MAXIT)GOTO 220                                1330
            PRINT 1003,MAXIT                                           1340
            GOTO 500                                                   1350
105   C                                                                1360
      C CHECK THE RUNNING TIME                                         1370
        220 IF(TTOUT.LT.TIME)GOTO 200                                  1380
            PRINT 1002,TIME                                            1390
            GOTO 500                                                   1400
110     300 NADD=0                                                     1410
            NDEL=0                                                     1420
            KOPT=0                                                     1430
            K2=0                                                       1440
            DO 301 N=1,NCENT                                           1450
```

```
      NN=KORIG(N)                                                       1460
      IF(NN.LE.0)GOTO 301                                               1470
C                                                                       1480
      K1=K2+1                                                           1490
      K2=NOPR(N)+K2                                                     1500
C                                                                       1510
      CALL REVPTH(NN,K1,K2,PTHFIL,NADD,NDEL,EPSI)                       1520
  301 CONTINUE                                                          1530
      NBPTH=NBPTH + NADD - NDEL                                         1540
      CALL OUTPTH(NPAIR,PTHFIL)                                         1550
      IF(NADD.GT.0)GOTO 200                                             1560
C                                                                       1570
C-------OPTIMUM IS REACHED OR TIME LIMIT OR CYCLE LIMIT                 1580
C                                                                       1590
  500 CONTINUE                                                          1600
      IPAGE =IPAGE + 1                                                  1610
      PRINT 1019,(RUNSEC(I),I=1,4)                                      1620
*                                                                       1630
C                                                                       1640
C--------SAVE SOLUTION VALUES                                          1650
      REWIND SOLFIL                                                     1660
      CALL DATE (TDAT)                                                  1670
      WRITE (SOLFIL,2019) IDEN,NUMCYC,TDAT                              1680
      DO 90 NO=1,NNODE                                                  1690
      J=NOLAB(NO)                                                       1700
      IF(J.LE.0)GOTO 90                                                 1710
      II=NO                                                             1720
   91 II=II+1                                                           1730
      JJ=NOLAB(II)-1                                                    1740
      IF(JJ.LE.0)GOTO 91                                                1750
      DO 92 N=J,JJ                                                      1760
C                                                                       1770
      PRINT 2020, NO,ND(N),VCUM(N),TRAV(N)                              1780
   92 CONTINUE                                                          1790
C                                                                       1800
   90 CONTINUE                                                          1810
C                                                                       1820
      REWIND ZNEFIL                                                     1830
      M=1                                                               1840
      K2=0                                                              1850
      DO 99 N=1,NCENT                                                   1860
      DO 93 I=1,NCENT                                                   1870
   93 ROW(I)=0                                                          1880
      IF(KORIG(N).LE.0)GOTO 96                                          1890
      K1=K2 + 1                                                         1900
      K2 = NOPR(N) + K2                                                 1910
      DO 94 J=K1,K2                                                     1920
      CALL READMS(PTHFIL,LINK,300,J)                                    1930
      ROW(LINK(4))=FCTDEM((LINK(1)/10000.),(LINK(2)/10000.))            1940
   94 CONTINUE                                                          1950
   96  PRINT 4001, (ROW(I),I=1,NCENT)                                   1960
   99 CONTINUE                                                          1970
*                                                                       1980
      ENDFILE SOLFIL                                                    1990
      ENDFILE ZNEFIL                                                    2000
*                                                                       2010
      RETURN                                                            2020
                                                                        2030
```

```
      SUBROUTINE OBFJN(RESVAL,DEMVAL,SYSVAL)                                2050
     *                                                                      2060
      COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                      2070
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)       2080
      COMMON /OD1/ NCENT, NPAIR, ROW(30), KORIG(30),NOPR(30)                2090
      COMMON MEMO1(900),MEMO2(550),SUBA(10)                                 2100
      DIMENSION LINK(300)                                                   2110
      INTEGER PTHFIL                                                        2120
      EQUIVALENCE (LINK(1),MEMO1(1))                                        2130
     *                                                                      2140
      RESVAL= DEMVAL= SYSVAL = 0.0                                          2150
      DO 10 I=1,NLINK                                                       2160
      RESVAL=RESVAL + TCOST(I,VCUM(I))                                      2170
   10 SYSVAL=SYSVAL + VCUM(I)*TRAV(I)                                       2180
      DO 40 J=1,NPAIR                                                       2190
      CALL READMS(PTHFIL,LINK,300,J)                                        2200
      UPQ=LINK(1)/10000.                                                    2210
      THETA=LINK(2)/10000.                                                  2220
      DEMVAL=DEMVAL+DEMINT(UPQ,THETA)                                       2230
   40 CONTINUE                                                              2240
     *                                                                      2250
      RETURN                                                                2260
      END                                                                   2270
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS

4	OBFJN

VARIABLES

SN	TYPE		RELOCATION	SN	TYPE		RELOCATION
0	CURNET	REAL	UNIT	0	DEMVAL	REAL	F.P.
77	I	INTEGER		100	J	INTEGER	
40	KORIG	INTEGER ARRAY	OD1	0	LINK	INTEGER ARRAY	/ /
0	MEMO1	INTEGER ARRAY	/ /	1604	MEMO2	INTEGER ARRAY	/ /
0	NCENT	INTEGER	OD1	374	ND	INTEGER ARRAY	NET1
1	NLINK	INTEGER	NET1	0	NNODE	INTEGER	NET1
2	NOLAB	INTEGER ARRAY	NET1	76	NOPR	INTEGER ARRAY	OD1
1	NPAIR	INTEGER	OD1	1	PTHFIL	INTEGER	UNIT
0	RESVAL	REAL	F.P.	2	ROW	REAL ARRAY	OD1
4	SOLFIL	REAL	UNIT	2552	SUBA	REAL ARRAY	/ /
0	SYSVAL	REAL	F.P.	2	TABIN	REAL	UNIT
102	THETA	REAL		1524	TRAV	REAL ARRAY	NET1
101	UPQ	REAL		2654	VCUM	REAL ARRAY	NET1
3	ZNEFIL	REAL	UNIT				

EXTERNALS

	TYPE	ARGS			
DEMINT	REAL	2		READMS	4
TCOST	REAL	2			

STATEMENT LABELS

0	10	0	40

```
      SUBROUTINE EQUIL (IPAIR,PTHFIL,KOPT,EPSI)                          3560
*                                                                        3570
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)    3580
      COMMON MEMO1(900),MEMO2(550),SUBA(10)                              3590
      DIMENSION LINK(300),PATH(50,5),IFLOW(5),LPJ(5)                     3600
      DIMENSION DELTA(600)                                               3610
      INTEGER PTHFIL,PATH                                                3620
      EQUIVALENCE (DELTA(1),MEMO1(301))                                  3630
      EQUIVALENCE (LINK(1),MEMO1(1)),(PATH(1,1),LINK(16)),               3640
     + (NO,LINK(3)),(NDEST,LINK(4)),(NP,LINK(5)),                        3650
     +       (IFLOW(1),LINK(6)),(LPJ(1),LINK(11))                        3660
      DIMENSION  DRT(6),X(6),U(6)                                        3670
      DIMENSION XX(6)                                                    3671
      LOGICAL FLAG                                                       3680
      REAL LAMBDA                                                        3690
      DATA GMAX/1000.0/                                                  3700
      DATA XLIM,ACCUR/1.0,1.0E-4/                                        3701
*                                                                        3710
*                                                                        3720
      KOPT=KOPT +1                                                       3750
      FLAG=.TRUE.                                                        3760
      LAMBDA=0.0                                                         3770
*                                                                        3780
      CALL SECOND(TTIN)                                                  3790
      CALL READMS (PTHFIL,LINK,300,IPAIR)                                3800
      IF(NP.EQ.0)RETURN                                                  3810
      J1=NP + 1                                                          3830
      THETA=LINK(2)/10000.                                               3840
      UPQ=LINK(1)/10000.                                                 3850
      DO 20 I=1,NP                                                       3870
   20 X(I)= IFLOW(I)/10000.0                                             3880
      NPA=J1                                                             3900
*                                                                        3910
*                                                                        3920
  100 DO 120 I=1,NP                                                      3930
      N=LPJ(I)                                                           3940
      SUM=0                                                              3950
      DO 121 J=1,N                                                       3960
  121 SUM=SUM + TRAV(PATH(J,I))                                          3970
  120 U(I)=SUM                                                           3980
*                                                                        3990
*                                                                        4000
      DO 140 I=1,J1                                                      4010
  140 DRT(I)=0                                                           4020
*                                                                        4030
*                                                                        4040
      UBA=1.0E11                                                         4050
      HMAX=0.0                                                           4060
      DO 150 I=1,NP                                                      4070
      IF(HMAX.GE.X(I)) GOTO 145                                          4080
      HMAX=X(I)                                                          4090
      IBA=I                                                              4100
  145 IF(UBA.LE.U(I)) GOTO 150                                           4110
      UBA=U(I)                                                           4120
  150 CONTINUE                                                           4130
      SUBA(IPAIR)=UBA                                                    4140
*                                                                        4150
```

```
      *
            S=S1=0.0                                                          4160
 60         ALPHA=0.0                                                         4170
            DO 160 I=1,NP                                                     4180
            IF(I.EQ.IBA) GOTO 160                                             4190
            D=U(IBA)-U(I)                                                     4200
            GAP=2.0*D/(U(IBA)+U(I)+0.001)                                     4210
 65         IF(ABS(GAP).LE.EPSI) GOTO 160                                     4220
            DRT(I)=D                                                          4230
            IF(D.LT.0.0) DRT(I)=D*X(I)                                        4240
            S=S+ABS(DRT(I))                                                   4250
            S1=S1+DRT(I)                                                      4260
 70         IF(D.GE.0.0) GOTO 160                                             4270
            IF(ALPHA.LT.-1.0*D) ALPHA=-1.0*D                                  4280
       160 CONTINUE                                                           4290
      *                                                                       4300
      *                                                                       4310
 75         D=UPQ-U(IBA)                                                      4320
            F=FPRIM(UPQ,THETA)                                                4330
            IF(ABS(F).LT.1.0) F=-1.0                                          4331
            DRT(J1)=F*D                                                       4332
            GAP=2.0*D/(U(IBA)+UPQ+0.001)                                      4333
 80         IF(ABS(GAP).LE.EPSI) DRT(J1)=0.0                                  4350
            IF(S.EQ.0.0.AND.DRT(J1).EQ.0.0) GOTO 500                          4360
            DRT(IBA)=(F*DRT(J1))-S1                                           4370
            IF(X(IBA).EQ.0.0.AND.DRT(IBA).LT.0.0) DRT(IBA)=0.0                4371
            IF(DRT(IBA).GE.0.0) GOTO 157                                      4372
 85         AA=-1.0*DRT(IBA)/X(IBA)                                           4373
            IF(ALPHA.LT.AA) ALPHA=AA                                          4374
       157 IF(DRT(J1).GE.0.0) GOTO 159                                        4375
            AA=-1.0*DRT(J1)                                                   4376
            IF(ALPHA.LT.AA) ALPHA=AA                                          4377
 90    159 IF(ALPHA.EQ.0.0) ALPHA=1.0                                         4378
            DO 158 I=1,NPA                                                    4379
       158 DRT(I)=DRT(I)/ALPHA                                                4380
      *                                                                       4381
      *                                                                       4430
 95   *                                                                       4440
      *                                                                       4510
            DO 175 I=1,NLINK                                                  4520
       175 DELTA(I)=0.0                                                       4580
            DO 182 I=1,NP                                                     4590
100         N=LPJ(I)                                                          4600
            D=DRT(I)                                                          4510
            IF(ABS(D).LE.0.0) GOTO 182                                        4620
            DO 180 J=1,N                                                      4630
       180 DELTA(PATH(J,I))=DELTA(PATH(J,I))+D                                4640
105    182 CONTINUE                                                           4650
            CALL GSM(LAMBDA,THETA,DRT(J1),UPQ,ACCUR,XLIM)                     4660
      *                                                                       4670
      *                                                                       4680
            IF(LAMBDA.LE.0.0) GOTO 1000                                       4690
110    190 CONTINUE                                                           4691
            DO 192 I=1,NP                                                     4700
            XX(I)=X(I)+LAMBDA*DRT(I)                                          4710
       192 CONTINUE                                                           4720
            ULDU=UPQ+LAMBDA*DRT(J1)                                           4730
                                                                              4731
```

```
      FULDU=FCTDEM(ULDU,THETA)                                          4732
      CALL NEWTON(XX(1),IBA,NP,FULDU)                                   4733
      IF(XX(IBA).GE.0.0) GOTO 203                                       4734
      LAMBDA=LAMBDA/2.0                                                 4735
      GOTO 190                                                          4736
  203 DRT(IBA)=XX(IBA)-X(IBA)                                           4737
      DO 204 I=1,NPA                                                    4738
      IF(I.EQ.IBA) GOTO 204                                             4739
      DRT(I)=DRT(I)*LAMBDA                                              4740
  204 CONTINUE                                                          4741
      FLAG=.F.                                                          4742
      DO 205 I=1,NLINK                                                  4780
      DELTA(I)=0.0                                                      4790
  205 CONTINUE                                                          4800
      DO 210 I=1,NP                                                     4810
      N=LPJ(I)                                                          4820
      D=DRT(I)                                                          4830
      IF(ABS(D).LE.0.0) GOTO 210                                        4840
      DO 215 J=1,N                                                      4850
  215 DELTA(PATH(J,I))=DELTA(PATH(J,I))+D                               4860
  210 CONTINUE                                                          4870
*                                                                       4880
*                                                                       4890
      DO 300 I=1,NLINK                                                  4900
      IF(ABS(DELTA(I)).LE.0.0) GOTO 300                                 4910
      VCUM(I)=VCUM(I)+DELTA(I)                                          4920
      IF(VCUM(I).LE.0.1) VCUM(I)=0.0                                    4930
      TRAV(I)=UCOST(I,VCUM(I))                                          4940
  300 CONTINUE                                                          4950
      DO 310 I=1,NP                                                     4960
      X(I)=X(I)+DRT(I)                                                  4961
      IF(X(I).LE.0.1) X(I)=0.0                                          4962
  310 CONTINUE                                                          4963
      UPQ=UPQ+DRT(J1)                                                   4980
      GOTO 100                                                          5000
*                                                                       5010
*                                                                       5020
*                                                                       5030
  500 IF(FLAG)GOTO 540                                                  5040
      KOPT=1                                                            5050
      DO 510 I=1,NP                                                     5060
      IFLOW(I)=X(I)*10000                                               5070
  510 CONTINUE                                                          5080
      LINK(1)=UPQ*10000                                                 5090
  540 CALL WRITMS(PTHFIL,LINK,300,IPAIR,1,1)                            5110
      RETURN                                                            5120
 1000 WRITE *," LAMBDA = 0"                                             5130
      PRINT 5000,(DRT(I),I=1,NPA)                                       5131
      PRINT 5000,(X(I),I=1,NP)                                          5132
      PRINT 5000,(DELTA(I),I=1,NLINK)                                   5133
      PRINT 5000,(VCUM(I),I=1,NLINK)                                    5134
 5000 FORMAT(7E15.8)                                                    5135
      CALL EXIT                                                         5136
      END                                                               5150
```

```
      SUBROUTINE GSM(LAMBDA,THETA,DU,U,ACCUR,XLIM)                      20
     *                                                                  30
     * SECTION DOREE                                                    40
     *                                                                  50
      COMMON /STAT/RUNSEC(10)                                           60
      EQUIVALENCE(SHSEC,RUNSEC(3))                                      70
      REAL LAMBDA                                                       80
     *                                                                  90
      CALL SECOND(TTIN)                                                100
     *                                                                 110
      TAU=(1.0+SQRT(5.))*.5                                            120
      G=0.0                                                            130
      D=XLIM                                                           140
      XB=D/TAU                                                         150
      XA=D-XB                                                          160
     *                                                                 170
      FXA=FCTVLU(XA,THETA,DU,U)                                        180
      FXB=FCTVLU(XB,THETA,DU,U)                                        190
     *                                                                 200
   77 IF(FXA-FXB) 80,80,78                                             210
   78 G=XA                                                             250
      XA=XB                                                            260
      XB=G+D-XA                                                        270
      FXA=FXB                                                          280
      FXB=FCTVLU(XB,THETA,DU,U)                                        290
      GOTO 82                                                          300
     *                                                                 310
   80 D=XB                                                             320
      XB=XA                                                           330
      XA=G+D-XB                                                        340
      FXB=FXA                                                          350
      FXA=FCTVLU(XA,THETA,DU,U)                                        360
     *                                                                 370
   82 IF(D-G-ACCUR)83,83,77                                            380
     *                                                                 390
   83 LAMBDA=XB                                                        400
      VAL=FXB                                                          410
      IF(FXB.LT.FXA) GOTO 90                                           420
      LAMBDA=XA                                                        430
   90 CALL SECOND(TTOUT)                                               440
      SHSEC=SHSEC+TTOUT-TTIN                                           450
      RETURN                                                           460
      END                                                              560
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 GSM

VARIABLES	SN	TYPE	RELOCATION				
0	ACCUR	REAL	F.P.	171	D	REAL	
0	DU	REAL	F.P.	174	FXA	REAL	
175	FXB	REAL		170	G	REAL	

```
      1              REAL FUNCTION FCTVLU(LAMBDA,THETA,DU,U)                                  1070
                     COMMON /LDATA/TO(600),BETA(600),LTYP(600)                                1080
                     COMMON /NET1/NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)          1090
      5              DIMENSION DELTA(600)                                                     1100
                     REAL LAMBDA,K2                                                           1110
                     EQUIVALENCE(DELTA(1),MEMO1(301))                                         1120
                     DATA GMAX,K2/1000.,0.15/                                                 1130
                    *                                                                         1140
     10             *                                                                         1150
                                                                                              1160
                     ULDU=U+(LAMBDA*DU)                                                       1170
                     F=0.0                                                                    1180
                     DO 10 I=1,NLINK                                                          1190
     15              VOL=VCUM(I)+(LAMBDA*DELTA(I))                                            1200
                     F=F+TCOST(I,VOL)                                                         1210
                  10 CONTINUE                                                                 1220
                     F=F+DEMINT(ULDU,THETA)                                                   1230
                     FCTVLU=F                                                                 1240
     20              RETURN                                                                   1250
                     END                                                                      1260
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
5 FCTVLU

VARIABLES	SN	TYPE	RELOCATION						
1130	BETA	REAL	ARRAY	LDATA	454	DELTA	REAL	ARRAY	/ /
0	DU	REAL		F.P.	66	F	REAL		
64	FCTVLU	REAL			62	GMAX	REAL		
67	I	INTEGER			63	K2	REAL		
0	LAMBDA	REAL		F.P.	2260	LTYP	INTEGER	ARRAY	LDATA
0	MEMO1	INTEGER	ARRAY	/ /	1604	MEMO2	INTEGER	ARRAY	/ /
374	ND	INTEGER	ARRAY	NET1	1	NLINK	INTEGER		NET1
0	NNODE	INTEGER		NET1	2	NOLAB	INTEGER	ARRAY	NET1
2652	SUBA	REAL	ARRAY	/ /	0	THETA	REAL		F.P.
0	TO	REAL	ARRAY	LDATA	1524	TRAV	REAL	ARRAY	NET1
0	U	REAL		F.P.	65	ULDU	REAL		
2654	VCUM	REAL	ARRAY	NET1	70	VOL	REAL		

EXTERNALS	TYPE	ARGS			
DEMINT	REAL	2	TCOST	REAL	2

STATEMENT LABELS
0 10

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
27	10	* I	13 16	13B	EXT REFS

COMMON BLOCKS	LENGTH
LDATA	1800
NET1	2052
/ /	1460

```
      1              SUBROUTINE NEWTON(X,IBA,NP,FULDU)                          8470
                     DIMENSION X(6)                                             8480
                     PT1=X(IBA)                                                 8490
                   1 SUM=0.                                                     8500
      5              DO 10 I=1,NP                                               8510
                     SUM=SUM+X(I)                                               8520
                  10 CONTINUE                                                   8530
                     SUM=SUM-FULDU                                              8540
                     X(IBA)=PT1-SUM                                             8550
     10              IF(ABS(X(IBA)-PT1).LT.0.001) RETURN                        8560
                     PT1=X(IBA)                                                 8570
                     GOTO 1                                                     8580
                     END                                                        8590
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 NEWTON

VARIABLES	SN	TYPE		RELOCATION				
0	FULDU	REAL		F.P.	62	I	INTEGER	
0	IBA	INTEGER		F.P.	0	NP	INTEGER	F.P.
60	PT1	REAL			61	SUM	REAL	
0	X	REAL	ARRAY	F.P.				

INLINE FUNCTIONS TYPE ARGS
ENTRIN

STATEMENT LABELS

A2.4. Programme DIRRES (direction réalisable, section 5.3)

```
      1              PROGRAM DIRRES(INPUT,OUTPUT,TAPE1,TAPE2,TAPE3,TAPE4,TAPE5)                    130
                     COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                              140
                     DIMENSION IDEN(10)                                                            150
                     INTEGER CURNET,TABIN,ZNEFIL,PTHFIL,SOLFIL                                     160
      5              DATA CURNET,PTHFIL,TABIN,ZNEFIL,SOLFIL                                        170
                    +    /1,2,3,4,5/                                                              180
                C                                                                                 190
                C                                                                                 200
                     READ 1004,(IDEN(J),J=1,8)                                                    210
     10              PRINT 1004,(IDEN(J),J=1,8)                                                   220
                1004 FORMAT (5X,7A8,A4)                                                           230
                C                                                                                 240
                     CALL TRAFIC(IDEN)                                                            250
                C                                                                                 260
     15              END                                                                          270
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
16377 DIRRES

VARIABLES	SN	TYPE	RELOCATION					
0	CURNET	INTEGER	UNIT	16426	IDEN	INTEGER	ARRAY	
16425	J	INTEGER		1	PTHFIL	INTEGER		UNIT
4	SOLFIL	INTEGER	UNIT	2	TABIN	INTEGER		UNIT
3	ZNEFIL	INTEGER	UNIT					

FILE NAMES								
0	INPUT	FMT	2043	OUTPUT	FMT	4106	TAPE1	6151 TAPE2
10214	TAPE3		12257	TAPE4		14322	TAPE5	

EXTERNALS	TYPE	ARGS
TRAFIC		1

STATEMENT LABELS
16423 1004 FMT

COMMON BLOCKS	LENGTH
UNIT	5

STATISTICS

PROGRAM LENGTH	538	43
BUFFER LENGTH	163658	7413
CM LABELED COMMON LENGTH	58	5

```
      SUBROUTINE TRAFIC (IDEN)                                          280
C                                                                       290
C                                                                       300
      COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                  310
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)   320
      COMMON /OD1/ NCENT, NPAIR, ROW(30), KORIG(30),NOPR(30)            330
      COMMON MEMO1(900),MEMO2(550),SUBA(10)                             340
      COMMON /RANDF/INDEX(800)                                          350
      COMMON /STAT/ RUNSEC(10)                                          360
      COMMON /PARAM/ EPSI,TIME, MAXIT,PRINT,DEMAND                      370
      DIMENSION IDEN(8), LINK(300)                                      380
      EQUIVALENCE (LINK(1),MEMO1(1))                                    390
      EQUIVALENCE (RDSEC,RUNSEC(1)),(SOLSEC,RUNSEC(2)),                 410
     +            (SHSEC,RUNSEC(3)),(PTHSEC,RUNSEC(4))                 420
      INTEGER CURNET,TABIN,ZNEFIL,PTHFIL,SOLFIL                         430
*                                                                       440
      DATA MAXPTH /400/                                                 460
      DATA EPSI,MAXIT,TIME /0.05,200,75.0/                              470
C                                                                       480
 1002 FORMAT(//1H0,20X,"6102I AFFECTATION TERMINEE.....DUREE (SECS)",   490
     +       " MAXIMUM  ", F10.0)                                       500
 1003 FORMAT(//1H0,20X,"6103I AFFECTATION TERMINEE ... ITERATION",      510
     +       " MAXIMUM  ",I8)                                           520
 1019 FORMAT ("0 SECS CPU UTILISE POUR - ",                             530
     +        "ENTRER LES DONNEES DU RESEAU",T60,"=",F10.2/             540
     +    T24," - ETABLIR LA SOLUTION INITIALE",T60,"=",F10.2/          550
     +    T24," - CALCULER LAMBDA (TOTALE)",T60,"=",F10.2/              560
     +    T24," - TROUVER LES CHEMINS PLUS COURTS",T60,"=",F10.2)       570
 2019 FORMAT (7A8,A4,1X,"IT=",I3,A10)                                   580
 2020 FORMAT(2I4,2F8.2)                                                 590
 2999 FORMAT(1H0,20X,                                                   600
     +          *F O N C T I O N   O B J E C T I V E *,                 610
     +/,3X,*ITER*,8X,*RESEAU*,15X,*DEMANDE*,10X,*   TOTALE*,11X,       620
     +*NOMBRE DE CHEMIN*,5X,*SEC. CUMULEES*/)                           630
 3000 FORMAT(3X,I3,3X,3(4X,E16.10),4X,I6,10X,F5.2)                      640
 4000 FORMAT(15F7.4)                                                    650
 4001 FORMAT(8F10.4)                                                    660
C                                                                       670
C  INITIALISATION                                                       680
C                                                                       690
      DO 10 I=1,10                                                      700
   10  RUNSEC(I)=0.0                                                    710
      CALL SECOND (TBEGIN)                                              720
      IPAGE =1                                                          730
      DEMVAL = 0.00                                                     750
C                                                                       760
C--------READ IN NETWORK                                                780
C                                                                       790
      CALL RDNET (CURNET,NCENT)                                         800
C                                                                       810
*------READ TRIP MATRX OR UPO MATRIX                                    820
*                                                                       830
      CALL RDOD (TABIN,NCENT,ZNEFIL,KORIG)                              840
   60 CALL SECOND(TTOUT)                                                850
      RDSEC=TTOUT-TBEGIN                                                860
      TTIN=TTOUT                                                        870
                                                                        880
```

```
      C
      C--------SET UP INITIAL (ALL OR NOTHING) SOLUTION                    890
   60 C                                                                    900
            REWIND ZNEFIL                                                 910
            CALL OPENMS (PTHFIL,INDEX,800,0,1)                            920
            DO 40 I=1,NLINK                                               930
            VCUM(I)=0.0                                                   940
   65    40 TRAV(I)=UCOST(I,VCUM(I))                                      950
            NPAIR=0                                                       960
            DO 42 N=1,NCENT                                               970
            NOPR(N)=0                                                     980
            NN=KORIG(N)                                                   990
   70       IF(NN.LE.0)GOTO 42                                           1000
            READ(ZNEFIL,4000) (ROW(J),J=1,NCENT)                         1010
            CALL LOADTR(NN,PTHFIL)                                       1020
            DO 43 I=1,NLINK                                              1030
         43    TRAV(I)=UCOST(I,VCUM(I))                                  1040
   75    42 CONTINUE                                                     1050
            NUMCYC = 0                                                   1060
            NBPTH=NPAIR                                                  1070
            CALL SECOND(TTOUT)                                           1080
            SOLSEC=TTOUT-TTIN                                            1090
   80 C                                                                  1100
            TTIN=TTOUT                                                   1110
      C                                                                  1120
      C--------PRINT HEADING FOR ITERATION LOG                           1130
        100 IPAGE = IPAGE + 1                                            1140
   85       PRINT 2999                                                   1150
      C                                                                  1160
            KOPT=0                                                       1170
        200 DO 205 J=1,NPAIR                                             1180
            CALL EQUIL (J,PTHFIL,KOPT,EPSI)                              1190
   90       IF(KOPT.EQ.NPAIR)GOTO 250                                    1200
        205 CONTINUE                                                     1210
        250 CONTINUE                                                     1220
      C                                                                  1230
            CALL OBFJN(RESVAL,DEMVAL,SYSVAL)                             1240
   95       CALL SECOND(TTOUT)                                           1250
            TTOUT=TTOUT-TBEGIN                                           1260
            NUMCYC=NUMCYC+1                                              1270
            TOTVAL=RESVAL+DEMVAL                                         1280
            PRINT 3000,NUMCYC,RESVAL,DEMVAL,TOTVAL,NBPTH,TTOUT           1290
  100       IF(KOPT.EQ.NPAIR)GOTO 300                                    1300
      C CHECK THE NUMBER OF ALLOWABLE CYCLES                             1310
        210 IF(NUMCYC.LT.MAXIT)GOTO 220                                  1320
            PRINT 1003,MAXIT                                             1330
            GOTO 500                                                     1340
  105 C                                                                  1350
      C CHECK THE RUNNING TIME                                           1360
        220 IF(TTOUT.LT.TIME)GOTO 200                                    1370
            PRINT 1002,TIME                                              1380
            GOTO 500                                                     1390
  110   300 NADD=0                                                       1400
            NDEL=0                                                       1410
            KOPT=0                                                       1420
            K2=0                                                         1430
            DO 301 N=1,NCENT                                             1440
                                                                         1450
```

```
      NN=KORIG(N)                                                       1460
      IF(NN.LE.0)GOTO 301                                               1470
C                                                                       1480
      K1=K2+1                                                           1490
      K2=NOPR(N)+K2                                                     1500
C                                                                       1510
      CALL REVPTH(NN,K1,K2,PTHFIL,NADD,NDEL,EPSI)                       1520
  301 CONTINUE                                                          1530
      NBPTH=NBPTH + NADD - NDEL                                         1540
      CALL OUTPTH(NPAIR,PTHFIL)                                         1550
      IF(NADD.GT.0)GOTO 200                                             1560
C                                                                       1570
C-------OPTIMUM IS REACHED OR TIME LIMIT OR CYCLE LIMIT                 1580
C                                                                       1590
  500 CONTINUE                                                          1600
      IPAGE =IPAGE + 1                                                  1610
      PRINT 1019,(RUNSEC(I),I=1,4)                                      1620
*                                                                       1630
C                                                                       1640
C--------SAVE SOLUTION VALUES                                          1650
      REWIND SOLFIL                                                     1660
      CALL DATE (TDAT)                                                  1670
      WRITE (SOLFIL,2019) IDEN,NUMCYC,TDAT                              1680
      DO 90 NO=1,NNODE                                                  1690
      J=NOLAB(NO)                                                       1700
      IF(J.LE.0)GOTO 90                                                 1710
      II=NO                                                             1720
   91 II=II+1                                                           1730
      JJ=NOLAB(II)-1                                                    1740
      IF(JJ.LE.0)GOTO 91                                                1750
      DO 92 N=J,JJ                                                      1760
C                                                                       1770
      PRINT 2020, NO,ND(N),VCUM(N),TRAV(N)                              1780
   92 CONTINUE                                                          1790
C                                                                       1800
   90 CONTINUE                                                          1810
C                                                                       1820
      REWIND ZNEFIL                                                     1830
      M=1                                                               1840
      K2=0                                                              1860
      DO 99 N=1,NCENT                                                   1870
      DO 93 I=1,NCENT                                                   1880
   93 ROW(I)=0                                                          1890
      IF(KORIG(N).LE.0)GOTO 96                                          1900
      K1=K2 + 1                                                         1910
      K2 = NOPR(N) + K2                                                 1920
      DO 94 J=K1,K2                                                     1930
      CALL READMS(PTHFIL,LINK,300,J)                                    1940
      ROW(LINK(4))=FCTDEM((LINK(1)/10000.),(LINK(2)/10000.))            1950
   94 CONTINUE                                                          1960
   96  PRINT 4001, (ROW(I),I=1,NCENT)                                   1970
   99 CONTINUE                                                          1980
*                                                                       1990
      ENDFILE SOLFIL                                                    2000
      ENDFILE ZNEFIL                                                    2010
*                                                                       2020
      RETURN                                                            2030
```

```
      SUBROUTINE OBFJN(RESVAL,DEMVAL,SYSVAL)                                2050
C                                                                           2060
      COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                      2070
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)       2080
      COMMON /OD1/ NCENT, NPAIR, ROW(30), KORIG(30),NOPR(30)                2090
      COMMON MEMO1(900),MEMO2(550),SUBA(10)                                 2100
      DIMENSION LINK(300)                                                   2110
      INTEGER PTHFIL                                                        2120
      EQUIVALENCE (LINK(1),MEMO1(1))                                        2130
C                                                                           2140
      RESVAL= DEMVAL= SYSVAL = 0.0                                          2150
      DO 10 I=1,NLINK                                                       2160
      RESVAL=RESVAL + TCOST(I,VCUM(I))                                      2170
   10 SYSVAL=SYSVAL + VCUM(I)*TRAV(I)                                       2180
      DO 40 J=1,NPAIR                                                       2190
      CALL READMS(PTHFIL,LINK,300,J)                                        2200
      UPQ=LINK(1)/10000.                                                    2210
      THETA=LINK(2)/10000.                                                  2220
      DEMVAL=DEMVAL+DEMINT(UPQ,THETA)                                       2230
   40 CONTINUE                                                              2240
C                                                                           2250
      RETURN                                                                2260
      END                                                                   2270
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 OBFJN

VARIABLES	SN	TYPE		RELOCATION
0	CURNET	REAL		UNIT
77	I	INTEGER		
40	KORIG	INTEGER	ARRAY	OD1
0	MEMO1	INTEGER	ARRAY	/ /
0	NCENT	INTEGER		OD1
1	NLINK	INTEGER		NET1
2	NOLAB	INTEGER	ARRAY	NET1
1	NPAIR	INTEGER		OD1
0	RESVAL	REAL		F.P.
4	SOLFIL	REAL		UNIT
0	SYSVAL	REAL		F.P.
102	THETA	REAL		
101	UPQ	REAL		
3	ZNEFIL	REAL		UNIT
0	DEMVAL	REAL		F.P.
100	J	INTEGER		
0	LINK	INTEGER	ARRAY	/ /
1604	MEMO2	INTEGER	ARRAY	/ /
374	ND	INTEGER	ARRAY	NET1
0	NNODE	INTEGER		NET1
76	NOPR	INTEGER	ARRAY	OD1
1	PTHFIL	INTEGER		UNIT
2	ROW	REAL	ARRAY	OD1
2652	SUBA	REAL	ARRAY	/ /
2	TABIN	REAL		UNIT
1524	TRAV	REAL	ARRAY	NET1
2654	VCUM	REAL	ARRAY	NET1

EXTERNALS	TYPE	ARGS			
DEMINT	REAL	2		READMS	4
TCOST	REAL	2			

STATEMENT LABELS
0 10 0 40

```
      SUBROUTINE EQUIL (IPAIR,PTHFIL,KOPT,EPSI)                          3560
*                                                                        3570
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)    3580
      COMMON MEMO1(900),MEMO2(550),SUBA(10)                              3590
      DIMENSION LINK(300),PATH(50,5),IFLOW(5),LPJ(5)                     3600
      DIMENSION DELTA(600)                                               3610
      INTEGER PTHFIL,PATH                                                3620
      EQUIVALENCE (DELTA(1),MEMO1(301))                                  3630
      EQUIVALENCE (LINK(1),MEMO1(1)),(PATH(1,1),LINK(16)),               3640
     + (NO,LINK(3)),(NDEST,LINK(4)),(NP,LINK(5)),                        3650
     +       (IFLOW(1),LINK(6)),(LPJ(1),LINK(11))                        3660
      DIMENSION  DRT(6),X(6),U(6)                                        3670
      LOGICAL FLAG                                                       3680
      LOGICAL FEQ                                                        3681
      REAL LAMBDA                                                        3690
      DATA GMAX/1000.0/                                                  3700
      DATA XLIM,ACCUR/1.0,1.0E-4/                                        3701
*                                                                        3710
*                                                                        3720
      KOPT=KOPT +1                                                       3750
      FLAG=.TRUE.                                                        3760
      LAMBDA=0.0                                                         3770
*                                                                        3780
      CALL SECOND(TTIN)                                                  3790
      CALL READMS (PTHFIL,LINK,300,IPAIR)                                3800
      IF(NP.EQ.0)RETURN                                                  3810
      J1=NP + 1                                                          3830
      THETA=LINK(2)/10000.                                               3840
      UPQ=LINK(1)/10000.                                                 3850
      DO 20 I=1,NP                                                       3870
   20 X(I)= IFLOW(I)/10000.0                                             3880
      NPA=J1                                                             3900
*                                                                        3910
*                                                                        3920
  100 DO 120 I=1,NP                                                      3930
      N=LPJ(I)                                                           3940
      SUM=0                                                              3950
      DO 121 J=1,N                                                       3960
  121 SUM=SUM + TRAV(PATH(J,I))                                          3970
  120 U(I)=SUM                                                           3980
*                                                                        3990
*                                                                        4000
      DO 140 I=1,J1                                                      4010
  140 DRT(I)=0                                                           4020
*                                                                        4030
*                                                                        4040
      UMAX=0.0                                                           4041
      UBA=1.0E11                                                         4050
      DO 150 I=1,NP                                                      4070
      IF(UBA.LE.U(I)) GOTO 145                                           4110
      IBA=I                                                              4111
      UBA=U(I)                                                           4120
  145 IF(UMAX.GT.U(I)) GOTO 150                                          4121
      IF(X(I).LE.0.1) GOTO 150                                           4122
      UMAX=U(I)                                                          4123
      IMAX=I                                                             4124
  150 CONTINUE                                                           4130
```

```
      SUBA(IPAIR)=UBA                                                    4140
     *                                                                   4150
60   *                                                                   4160
      FEQ=.FALSE.                                                        4162
      GAP=2.0*ABS(UPQ-UBA)/(UPQ+UBA+EPSI)                                4163
      IF(GAP.GT.EPSI) GOTO 200                                           4164
      IF(UMAX.EQ.0.0) GOTO 500                                           4165
65    GAP=2.0*(UMAX-UBA)/(UMAX+UBA+EPSI)                                 4166
      IF(GAP.LT.EPSI) GOTO 500                                           4167
      FEQ=.TRUE.                                                         4168
      GOTO 156                                                           4169
  200 IF(UPQ.GT.UBA) GOTO 157                                            4181
70    156 IF(NP.EQ.1) GOTO 157                                           4182
      DO 155 I=1,NP                                                      4183
      DRT(I)=-X(I)                                                       4184
  155 CONTINUE                                                           4185
  157 IF(.NOT.FEQ) GOTO 160                                              4186
75    DRT(IBA)=FCTDEM(UPQ,THETA)-X(IBA)                                  4187
      XLIM=1.0                                                           4188
      GOTO 165                                                           4189
  160 DRT(J1)=FPRIM(UPQ,THETA)*(UPQ-UBA)                                 4190
      IF(DRT(J1).GE.0.0) GOTO 162                                        4191
80    XLIM=-UPQ/DRT(J1)                                                  4192
      GOTO 165                                                           4194
  162 D=FCTDEM(UPQ,THETA)-FCTDEM(UPQ+DRT(J1),THETA)                      4195
      DO 164 I=1,NP                                                      4196
      IF(I.EQ.IBA) GOTO 164                                              4197
85    D=D-X(I)                                                           4198
  164 CONTINUE                                                           4199
      XLIM=1.0                                                           4200
      IF(D.LT.0.0) GOTO 165                                              4203
      XLIM=X(IBA)/D                                                      4204
90    IF(XLIM.EQ.0.0) XLIM=1.0                                           4205
      IF(XLIM.GT.1.0.AND.NP.GT.1) XLIM=1.0                               4206
  165 CONTINUE                                                           4207
      CALL GSM(LAMBDA,THETA,ACCUR,X(1),DRT(1),UPQ,UBA,FEQ,IBA,XLIM)      4210
      DO 170 I=1,NPA                                                     4330
95    DRT(I)=DRT(I)*LAMBDA                                               4340
  170  CONTINUE                                                          4360
      IF(FEQ) GOTO 179                                                   4370
      UU=UPQ+DRT(J1)                                                     4380
  172 DRT(IBA)=FCTDEM(UU,THETA)-FCTDEM(UPQ,THETA)                        4400
100   IF(UPQ.GT.UBA) GOTO 178                                            4401
      D=0.0                                                              4402
      DO 175 I=1,NP                                                      4403
      IF(I.EQ.IBA) GOTO 175                                              4404
      D=D+(LAMBDA*X(I))                                                  4405
105   175 CONTINUE                                                       4406
      DRT(IBA)=DRT(IBA)+D                                                4407
  178 IF(DRT(IBA).LT.0.0.AND.X(IBA).EQ.0.0) DRT(IBA)=0.0                 4417
     *                                                                   4450
     *                                                                   4480
110   179 FLAG=.F.                                                       4490
      DO 205 I=1,NLINK                                                   4500
  205 DELTA(I)=0.0                                                       4510
      DO 210 I=1,NP                                                      4520
      N=LPJ(I)                                                           4530
```

```
                      D=DRT(I)                                                  4540
                      IF(D.EQ.0.0) GOTO 210                                     4550
                      DO 208 J=1,N                                              4560
                  208 DELTA(PATH(J,I))=DELTA(PATH(J,I))+D                       4570
                  210 CONTINUE                                                  4580
                  *                                                             4880
                  *                                                             4890
                      DO 300 I=1,NLINK                                          4900
                      IF(ABS(DELTA(I)).LE.0.0) GOTO 300                         4910
                      VCUM(I)=VCUM(I)+DELTA(I)                                  4920
                      IF(VCUM(I).LE.0.1) VCUM(I)=0.0                            4930
                      TRAV(I)=UCOST(I,VCUM(I))                                  4940
                  300 CONTINUE                                                  4950
                      DO 310 I=1,NP                                             4960
                      X(I)=X(I)+DRT(I)                                          4961
                      IF(X(I).LE.0.1) X(I)=0.0                                  4962
                  310 CONTINUE                                                  4963
                      UPQ=UPQ+DRT(J1)                                           4980
                      GOTO 100                                                  5000
                  *                                                             5010
                  *                                                             5020
                  *                                                             5030
                  500 IF(FLAG)GOTO 540                                          5040
                      KOPT=1                                                    5050
                      DO 510 I=1,NP                                             5060
                      IFLOW(I)=X(I)*10000                                       5070
                  510 CONTINUE                                                  5080
                      LINK(I)=UPQ*10000                                         5090
                  540 CALL WRITMS(PTHFIL,LINK,300,IPAIR,1,1)                    5110
                      RETURN                                                    5120
                      END                                                       5150
```

CARD NR. SEVERITY DETAILS DIAGNOSIS OF PROBLEM

CARD NR.	SEVERITY	DETAILS	DIAGNOSIS OF PROBLEM
118	I	NP	THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.
140	I	NP	THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 EQUIL

VARIABLES	SN	TYPE	RELOCATION				
577	ACCUR	REAL					
454	DELTA	REAL	ARRAY	/ / F.P.			
0	EPSI	REAL					
610	FLAG	LOGICAL					
575	GMAX	REAL					
626	IBA	INTEGER					
627	IMAX	INTEGER					
631	D	REAL					
633	DRT	REAL	ARRAY				
611	FEQ	LOGICAL					
630	GAP	REAL					
617	I	INTEGER					
5	IFLOW	INTEGER	ARRAY	/ / F.P.			
0	IPAIR	INTEGER					

```
      SUBROUTINE GSM(LAMBDA,THETA,ACCUR,X,DRT,U,UBA,FEQ,IBA,XLIM)
*
* METHODE DE LA SECTION DOREE
*
      COMMON /NET1/NNODE,NLINK,NOLAB(250),ND(600),TRAV(500),VCUM(600)
      COMMON /STAT/RUNSEC(10)
      DIMENSION DRT(6),X(6)
      EQUIVALENCE(SHSEC,RUNSEC(3))
      COMMON MEMO1(900),MEMO2(550)
      DIMENSION DELTA(600),PATH(50,5) ,LPJ(5),LINK(300)
      INTEGER PATH
      EQUIVALENCE(MEMO1(301),DELTA(1))
      EQUIVALENCE (MEMO1(1),LINK(1)),(NP,LINK(5)),(PATH(1,1),LINK(16)),(
     +LPJ(1),LINK(11))
      LOGICAL FEQ
      REAL LAMBDA
      J1=NP+1
      DU=DRT(J1)
      CALL SECOND(TTIN)
      TAU=(1.0+SQRT(5.))*.5
*
      G=0.0
      D=XLIM
    1 XB=D/TAU
      XA=D-XB
*
      NPA=NP+1
      IF(.NOT.FEQ) GOTO 60
      DO 3500 I=1,NLINK
 3500 DELTA(I)=0.0
      DO 5000 I=1,NP
      N=LPJ(I)
      DD=DRT(I)
      IF(DD.EQ.0.0) GOTO 5000
      DO 4000 J=1,N
      IX=PATH(J,I)
 4000 DELTA(IX)=DELTA(IX)+DD
 5000 CONTINUE
*
   60 FXA=FCTVLU(XA,THETA,U,UBA,DRT(1),IBA,FEQ,X)
      FXB=FCTVLU(XB,THETA,U,UBA,DRT(1),IBA,FEQ,X)
*
   77 IF(FXA-FXB) 80,80,78
   78 G=XA
      XA=XB
      XB=G+D-XA
      FXA=FXB
      FXB=FCTVLU(XB,THETA,U,UBA,DRT(1),IBA,FEQ,X)
      GOTO 82
*
   80 D=XB
      XB=XA
      XA=G+D-XB
      FXB=FXA
      FXA=FCTVLU(XA,THETA,U,UBA,DRT(1),IBA,FEQ,X)
*
   82 IF(D-G-ACCUR) 83,83,77
```

```
            *                                                          610
                83 LAMBDA=XB                                           
  60                VAL=FXB                                            620
                    IF(FXB.LT.FXA) GOTO 1000                           630
                    VAL=FXA                                            640
                    LAMBDA=XA                                          650
            *                                                          660
  65          1000 CALL SECOND(TTOUT)                                  670
                    SHSEC=SHSEC+TTOUT-TTIN                             680
                    RETURN                                             690
                    END                                                700
                                                                       710
```

```
CARD NR. SEVERITY  DETAILS   DIAGNOSIS OF PROBLEM

    37      I        NP      THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.
```

```
      SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    4  GSM

VARIABLES     SN  TYPE              RELOCATION
     0  ACCUR      REAL                  F.P.      324  D        REAL
   332  DD         REAL                            454  DELTA    REAL      ARRAY   / /
     0  DRT        REAL      ARRAY       F.P.      320  DU       REAL
     0  FEQ        LOGICAL               F.P.      335  FXA      REAL
   336  FXB        REAL                            323  G        REAL
   330  I          INTEGER                           0  IBA      INTEGER           F.P.
   334  IX         INTEGER                         333  J        INTEGER
   317  J1         INTEGER                           0  LAMBDA   REAL              F.P.
     0  LINK       INTEGER   ARRAY       / /        12  LPJ      INTEGER   ARRAY   / /
     0  MEMO1      INTEGER   ARRAY       / /      1604  MEMO2    INTEGER   ARRAY   / /
   331  N          INTEGER                         374  ND       INTEGER   ARRAY   NET1
     1  NLINK      INTEGER               NET1        0  NNODE    INTEGER           NET1
     2  NOLAB      INTEGER   ARRAY       NET1        4  NP       INTEGER           / /
   327  NPA        INTEGER                          17  PATH     INTEGER   ARRAY   / /
     0  RUNSEC     REAL      ARRAY       STAT        2  SHSEC    REAL              STAT
   322  TAU        REAL                              0  THETA    REAL              F.P.
  1524  TRAV       REAL      ARRAY       NET1      321  TTIN     REAL
   340  TTOUT      REAL                              0  U        REAL              F.P.
     0  UBA        REAL                  F.P.      337  VAL      REAL
  2654  VCUM       REAL      ARRAY       NET1        0  X        REAL      ARRAY   F.P.
   326  XA         REAL                            325  XB       REAL
     0  XLIM       REAL                  F.P.

EXTERNALS         TYPE   ARGS
     FCTVLU       REAL    8                      SECOND            1
     SQRT         REAL    1 LIBRARY
```

```
      REAL FUNCTION FCTVLU(LAMBDA,THETA,U,UBA,DRT,IBA,FEQ,X)                    720
      COMMON /LDATA/TO(600),BETA(600),LTYP(600)                                 730
      COMMON /NET1/NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)           740
      COMMON MEMO1(900),MEMO2(550),SUBA(10)                                     750
      DIMENSION DELTA(600),X(6)                                                 760
      REAL LAMBDA,K2                                                            770
      LOGICAL FEQ                                                               780
      DIMENSION DRT(6),LINK(300),XDRT(6),PATH(50,5),LPJ(5)                      790
      INTEGER PATH                                                              800
      EQUIVALENCE(MEMO1(1),LINK(1)),(NP,LINK(5)),(PATH(1,1),LINK(16)),(L        810
     +PJ(1),LINK(11))                                                           820
      EQUIVALENCE(DELTA(1),MEMO1(301))                                          830
      DATA GMAX,K2/1000.,0.15/                                                  840
*                                                                               850
*                                                                               860
      NPA=NP+1                                                                  870
      J1=NPA                                                                    880
      DU=DRT(J1)                                                                890
      IF(.NOT.FEQ) GOTO 100                                                     900
      F=0.0                                                                     920
      DO 10 I=1,NLINK                                                           930
      VOL=VCUM(I)+(LAMBDA*DELTA(I))                                             940
      F=F+TCOST(I,VOL)                                                          950
   10 CONTINUE                                                                  960
      FCTVLU=F                                                                  980
      RETURN                                                                    990
  100 DO 120 I=1,NPA                                                           1000
  120 XDRT(I)=DRT(I)*LAMBDA                                                    1010
      UU=U+XDRT(J1)                                                            1011
      XDRT(IBA)=FCTDEM(UU,THETA)-FCTDEM(U,THETA)                               1012
      IF(U.GT.UBA) GOTO 135                                                    1013
      D=0.0                                                                    1014
      DO 130 I=1,NP                                                            1015
      IF(I.EQ.IBA) GOTO 130                                                    1016
      D=D+(LAMBDA*X(I))                                                        1017
  130 CONTINUE                                                                 1018
      XDRT(IBA)=XDRT(IBA)+D                                                    1019
  135 IF(XDRT(IBA).LT.0.0.AND.X(IBA).EQ.0.0) XDRT(IBA)=0.0                     1029
*                                                                              1070
  140 CONTINUE                                                                 1080
      DO 155 I=1,NLINK                                                         1110
  155 DELTA(I)=0.0                                                             1120
      DO 170 I=1,NP                                                            1130
      N=LPJ(I)                                                                 1140
      D=XDRT(I)                                                                1150
      IF(D.EQ.0.0) GOTO 170                                                    1160
      DO 165 J=1,N                                                             1170
      IX=PATH(J,I)                                                             1180
  165 DELTA(IX)=DELTA(IX)+D                                                    1190
  170 CONTINUE                                                                 1200
      F=0.0                                                                    1210
      DO 200 I=1,NLINK                                                         1220
      VOL=VCUM(I)+DELTA(I)                                                     1230
  200 F=F+TCOST(I,VOL)                                                         1240
      UU=U+XDRT(J1)                                                            1241
      F=F+DEMINT(UU,THETA)                                                     1242
      FCTVLU=F                                                                 1251
```

A2.5 Programme ETADB (algorithme B1, section 6.3)

```
      PROGRAM ETABD      73/74   OPT=0 TRACE                    FTN 4.6+428      07/04/78  13.12.27      PAGE     1

 1                PROGRAM ETABD(INPUT,OUTPUT,TAPE1,TAPE2,TAPE3,TAPE4,TAPE5)                    130
                  *                                                                            140
                  * AFFECTATION DU TRAFIC AVEC DEMANDE ELASTIQUE SANS INVERSE                   150
                  * METHODE DE BENDERS GENERALISEE                                            160
 5                *                                                                            170
                       COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                       180
                       DIMENSION IDEN(10)                                                     190
                       INTEGER CURNET,TABIN,ZNEFIL,PTHFIL,SOLFIL                              200
                       DATA CURNET,PTHFIL,TABIN,ZNEFIL,SOLFIL                                 210
10                   +    /1,2,3,4,5/                                                         220
                  *                                                                           230
                  *                                                                           240
                       READ 1004,(IDEN(J),J=1,8)                                              250
                       PRINT 1004,(IDEN(J),J=1,8)                                             260
15                1004 FORMAT (5X,7A8,A4)                                                    270
                  *                                                                           280
                       CALL TRAFIC(IDEN)                                                      290
                  *                                                                           300
                       END                                                                    310


      SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
16377  ETABD

VARIABLES     SN  TYPE            RELOCATION
    0  CURNET     INTEGER             UNIT          16426  IDEN      INTEGER    ARRAY
16425  J          INTEGER                               1  PTHFIL    INTEGER             UNIT
    4  SOLFIL     INTEGER             UNIT              2  TABIN     INTEGER             UNIT
    3  ZNEFIL     INTEGER             UNIT

FILE NAMES       MODE
    0  INPUT      FMT          2043  OUTPUT    FMT          4106  TAPE1                6151  TAPE2
10214  TAPE3                  12257  TAPE4                 14322  TAPE5

EXTERNALS          TYPE   ARGS
      TRAFIC                   1

STATEMENT LABELS
16423  1004     FMT

COMMON BLOCKS   LENGTH
      UNIT           5

STATISTICS
  PROGRAM LENGTH                  538      43
  BUFFER LENGTH                163658    7413
  CM LABELED COMMON LENGTH         58       5
```

```
      SUBROUTINE TRAFIC (IDEN)                                          320
C                                                                       330
C                                                                       340
C                                                                       350
      COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                  360
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)   370
      COMMON /OD1/ NCENT, NPAIR, ROW(30), KORIG(30),NOPR(30)            380
      COMMON MEMO1(900),MEMO2(550),SUBA(10)                             390
      COMMON /RANDF/INDEX(800)                                          400
      COMMON /STAT/ RUNSEC(10)                                          410
      COMMON /PARAM/ EPSI,TIME, MAXIT,PRINT,DEMAND                      420
      DIMENSION IDEN(8), LINK(300)                                      430
      EQUIVALENCE (LINK(1),MEMO1(1))                                    440
      EQUIVALENCE (RDSEC,RUNSEC(1)),(SOLSEC,RUNSEC(2)),                 450
     +            (SHSEC,RUNSEC(3)),(PTHSEC,RUNSEC(4))                 460
      INTEGER CURNET,TABIN,ZNEFIL,PTHFIL,SOLFIL                         470
*                                                                       480
      DATA MAXPTH /400/                                                 490
      DATA EPSI,MAXIT,TIME /0.001,200,75.0/                             500
C                                                                       510
 1002 FORMAT(//1H0,20X,"6102I AFFECTATION TERMINEE.....DUREE (SECS)",   520
     +       " MAXIMUM :", F10.0)                                       530
 1003 FORMAT(//1H0,20X,"6103I AFFECTATION TERMINEE ... ITERATION",      540
     +       " MAXIMUM :",I8)                                           550
 1019 FORMAT ("0 SECS CPU UTILISE POUR - ",                             560
     +        "ENTRER LES DONNEES DU RESEAU",T70,"=",F10.2/             570
     +    T24," - ETABLIR LA SOLUTION INITIALE",T70,"=",F10.2/          580
     +    T24," - CALCULER LAMBDA (TOTALE)",T70,"=",F10.2/              590
     +    T24," - TROUVER LES CHEMINS PLUS COURTS",T70,"=",F10.2/       600
     +T24," - SOLUTIONNER MASTER PROBLEM (TOT.)",T70,"=",F10.2)         601
 2019 FORMAT (7A8,A4,1X,"IT=",I3,A10)                                   610
 2020 FORMAT(2I4,2F8.2)                                                 620
 2999 FORMAT(1H0,20X,                                                   630
     +          "F O N C T I O N   O B J E C T I V E ",                 640
     +/,3X,"ITER",8X,"RESEAU",15X,"DEMANDE",10X,"   TOTALE",11X,       650
     +"NOMBRE DE CHEMIN",5X,"SEC. CUMULEES"/)                           660
 3000 FORMAT(3X,I3,3X,3(4X,E16.10),4X,I6,10X,F5.2)                      670
 4000 FORMAT(15F7.4)                                                    680
 4001 FORMAT(8F10.4)                                                    690
C                                                                       700
C  INITIALISATION                                                       710
C                                                                       720
      DO 10 I=1,10                                                      730
   10  RUNSEC(I)=0.0                                                    740
      CALL SECOND (TBEGIN)                                              750
      IPAGE =1                                                          760
      DEMVAL = 0.00                                                     770
C                                                                       780
C--------READ IN NETWORK                                                790
C                                                                       800
      CALL RDNET (CURNET,NCENT)                                         810
C                                                                       820
*------READ TRIP MATRX OR UPO MATRIX                                    830
*                                                                       840
      CALL RDOD (TABIN,NCENT,ZNEFIL,KORIG)                              850
   60 CALL SECOND(TTOUT)                                                860
      RDSEC=TTOUT-TBEGIN                                                870
```

```
                TTIN=TTOUT                                                      880
              C                                                                 890
   60         C--------SET UP INITIAL (ALL OR NOTHING) SOLUTION                 900
              C                                                                 910
                REWIND ZNEFIL                                                   920
                CALL OPENMS (PTHFIL,INDEX,800,0,1)                              930
                DO 40 I=1,NLINK                                                 940
   65           VCUM(I)=0.0                                                     950
             40 TRAV(I)=UCOST(I,VCUM(I))                                        960
                NPAIR=0                                                         970
                DO 42 N=1,NCENT                                                 980
                NOPR(N)=0                                                       990
   70           NN=KORIG(N)                                                    1000
                IF(NN.LE.0)GOTO 42                                             1010
                READ(ZNEFIL,4000) (ROW(J),J=1,NCENT)                           1020
                CALL LOADTR(NN,PTHFIL)                                         1030
                DO 43 I=1,NLINK                                                1040
   75        43     TRAV(I)=UCOST(I,VCUM(I))                                  1050
             42 CONTINUE                                                       1060
                NUMCYC = 0                                                     1070
                NBPTH=NPAIR                                                    1080
                CALL SECOND(TTOUT)                                             1090
   80           SOLSEC=TTOUT-TTIN                                              1100
              C                                                                1110
                TTIN=TTOUT                                                     1120
              C                                                                1130
              C--------PRINT HEADING FOR ITERATION LOG                        1140
   85       100 IPAGE = IPAGE + 1                                             1150
                PRINT 2999                                                     1160
              C                                                                1170
                KOPT=0                                                         1180
            200 DO 205 J=1,NPAIR                                               1190
   90           CALL GB (J,PTHFIL,KOPT,EPSI)                                  1200
                IF(KOPT.EQ.NPAIR)GOTO 250                                      1210
            205 CONTINUE                                                       1220
            250 CONTINUE                                                       1230
              C                                                                1240
   95           CALL OBFJN(RESVAL,DEMVAL,SYSVAL)                              1250
                CALL SECOND(TTOUT)                                             1260
                TTOUT=TTOUT-TBEGIN                                             1270
                NUMCYC=NUMCYC+1                                                1280
                TOTVAL=RESVAL+DEMVAL                                           1290
  100           PRINT 3000,NUMCYC,RESVAL,DEMVAL,TOTVAL,NBPTH,TTOUT            1300
                IF(KOPT.EQ.NPAIR)GOTO 300                                      1310
              C CHECK THE NUMBER OF ALLOWABLE CYCLES                           1320
            210 IF(NUMCYC.LT.MAXIT)GOTO 220                                    1330
                PRINT 1003,MAXIT                                               1340
  105           GOTO 500                                                       1350
              C                                                                1360
              C CHECK THE RUNNING TIME                                         1370
             220  IF(TTOUT.LT.TIME)GOTO 200                                   1380
                PRINT 1002,TIME                                                1390
  110           GOTO 500                                                       1400
            300 CALL OUTPTH(NPAIR,PTHFIL)                                      1410
              C                                                                1570
              C-------OPTIMUM IS REACHED OR TIME LIMIT OR CYCLE LIMIT           1580
              C                                                                1590
```

```
      500 CONTINUE                                                      1600
          IPAGE =IPAGE + 1                                              1610
          PRINT 1019,(RUNSEC(I),I=1,5)                                  1620
      *                                                                 1630
      C                                                                 1640
      C---------SAVE SOLUTION VALUES                                    1650
          REWIND SOLFIL                                                 1660
          CALL DATE (TDAT)                                              1670
          WRITE (SOLFIL,2019) IDEN,NUMCYC,TDAT                          1680
          DO 90 NO=1,NNODE                                              1690
          J=NOLAB(NO)                                                   1700
          IF(J.LE.0)GOTO 90                                             1710
          II=NO                                                         1720
       91 II=II+1                                                       1730
          JJ=NOLAB(II)-1                                                1740
          IF(JJ.LE.0)GOTO 91                                            1750
          DO 92 N=J,JJ                                                  1760
      C                                                                 1770
          PRINT 2020, NO,ND(N),VCUM(N),TRAV(N)                          1780
       92 CONTINUE                                                      1790
      C                                                                 1800
       90 CONTINUE                                                      1810
      C                                                                 1820
          REWIND ZNEFIL                                                 1830
          M=1                                                           1840
          K2=0                                                          1850
          DO 99 N=1,NCENT                                               1860
          DO 93 I=1,NCENT                                               1870
       93 ROW(I)=0                                                      1880
          IF(KORIG(N).LE.0)GOTO 96                                      1890
          K1=K2 + 1                                                     1900
          K2 = NOPR(N) + K2                                             1910
          DO 94 J=K1,K2                                                 1920
          CALL READMS(PTHFIL,LINK,300,J)                                1930
          ROW(LINK(4))=LINK(300)/10000.                                 1940
       94 CONTINUE                                                      1950
       96  PRINT 4001, (ROW(I),I=1,NCENT)                              1960
       99 CONTINUE                                                      1970
      *                                                                 1980
          ENDFILE SOLFIL                                                1990
          ENDFILE ZNEFIL                                                2000
      *                                                                 2010
          RETURN                                                        2020
          END                                                           2030
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 TRAFIC

VARIABLES	SN	TYPE	RELOCATION				
			UNIT				
0 CURNET		INTEGER		4 DEMAND	REAL	PARAM	
641 DEMVAL		REAL		0 EPSI	REAL	PARAM	

```
      SUBROUTINE OBFJN(RESVAL,DEMVAL,SYSVAL)                                2040
     *                                                                      2050
      COMMON /UNIT/ CURNET, PTHFIL,TABIN,ZNEFIL,SOLFIL                      2060
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)       2070
      COMMON /OD1/ NCENT, NPAIR, ROW(30), KORIG(30),NOPR(30)                2080
      COMMON MEMO1(900),MEMO2(550),SUBA(10)                                 2090
      DIMENSION LINK(300)                                                   2100
      INTEGER PTHFIL                                                        2110
      EQUIVALENCE (LINK(1),MEMO1(1))                                        2120
     *                                                                      2130
      RESVAL= DEMVAL= SYSVAL = 0.0                                          2140
      DO 10 I=1,NLINK                                                       2150
      RESVAL=RESVAL + TCOST(I,VCUM(I))                                      2160
   10 SYSVAL=SYSVAL + VCUM(I)*TRAV(I)                                       2170
      DO 40 J=1,NPAIR                                                       2180
      CALL READMS(PTHFIL,LINK,300,J)                                        2190
      UPQ=LINK(1)/10000.                                                    2200
      THETA=LINK(2)/10000.                                                  2210
      DEMVAL=DEMVAL+DEMINT(UPQ,THETA)                                       2220
   40 CONTINUE                                                              2230
     *                                                                      2240
      RETURN                                                                2250
      END                                                                   2260
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 OBFJN

VARIABLES	SN	TYPE		RELOCATION					
0	CURNET	REAL		UNIT	0	DEMVAL	REAL		F.P.
77	I	INTEGER			100	J	INTEGER		
40	KORIG	INTEGER	ARRAY	OD1	0	LINK	INTEGER	ARRAY	/ /
0	MEMO1	INTEGER	ARRAY	/ /	1604	MEMO2	INTEGER	ARRAY	/ /
0	NCENT	INTEGER		OD1	374	ND	INTEGER	ARRAY	NET1
1	NLINK	INTEGER		NET1	0	NNODE	INTEGER		NET1
2	NOLAB	INTEGER	ARRAY	NET1	76	NOPR	INTEGER	ARRAY	OD1
1	NPAIR	INTEGER		OD1	1	PTHFIL	INTEGER		UNIT
0	RESVAL	REAL		F.P.	2	ROW	REAL	ARRAY	OD1
4	SOLFIL	REAL		UNIT	2652	SUBA	REAL	ARRAY	/ /
0	SYSVAL	REAL		F.P.	2	TABIN	REAL		UNIT
102	THETA	REAL			1524	TRAV	REAL	ARRAY	NET1
101	UPQ	REAL			2654	VCUM	REAL	ARRAY	NET1
3	ZNEFIL	REAL		UNIT					

EXTERNALS	TYPE	ARGS			
DEMINT	REAL	2		READMS	4
TCOST	REAL	2			

STATEMENT LABELS
0 10 0 40

```
      SUBROUTINE EQFIX(EPSI,FUPQ,UBA,VAL,F,IPAIR,PTHFIL)                3480
*                                                                       3490
      COMMON/NET1/ MNODE,NLINK,NOLAB(250),MD(600),TRAV(600),VCUM(600)   3500
      COMMON MEMO1(900),MEMO2(550)                                      3510
      DIMENSION LINK(300),PATH(50,5),IFLOW(5),LPJ(5)                    3520
      DIMENSION DELTA(100)                                              3530
      INTEGER PATH                                                      3540
      EQUIVALENCE (DELTA(1),MEMO1(301))                                 3550
      EQUIVALENCE (LINK(1),MEMO1(1)),(PATH(1,1),LINK(16)),              3560
     + (NO,LINK(3)),(NDEST,LINK(4)),(NP,LINK(5)),                       3570
     +       (IFLOW(1),LINK(6)),(LPJ(1),LINK(11))                       3580
      DIMENSION  DRT(6),X(6),U(6)                                       3590
      DIMENSION XX(6)                                                   3591
      LOGICAL F                                                         3600
      INTEGER PTHFIL                                                    3601
      REAL LAMBDA                                                       3610
      DATA GMAX/2000.0/                                                 3620
      DATA ACCUR,XLIM/1.0E-4,1.0/                                       3621
*                                                                       3630
*                                                                       3640
      F=.TRUE.                                                          3660
      LAMBDA=0.0                                                        3670
*                                                                       3680
      CALL SECOND(TTIN)                                                 3690
      THETA=LINK(2)/10000.                                              3720
      DO 20 I=1,NP                                                      3740
   20 X(I)= IFLOW(I)/10000.0                                            3750
      X(6)=LINK(300)/10000.                                             3751
      XXX=FUPQ                                                          3752
      DIFF=XXX-X(6)                                                     3753
      GAP=2.0*DIFF/(XXX+X(6)+0.001)                                     3754
      IF(ABS(GAP).LT.0.01) GOTO 100                                     3755
      DO 21 I=1,NP                                                      3756
   21 XX(I)=0.0                                                         3757
      XX(1)=XXX-X(1)                                                    3758
      X(1)=XXX                                                          3759
      IF(NP.LT.2) GOTO 31                                               3760
      DO 25 I=2,NP                                                      3761
      XX(I)=-X(I)                                                       3762
      X(I)=0.0                                                          3763
   25 CONTINUE                                                          3764
   31 DO 60 I=1,NP                                                      3771
      IF(XX(I).EQ.0.0) GOTO 60                                          3772
      N=LPJ(I)                                                          3773
      DO 50 J=1,N                                                       3774
      IX=PATH(J,I)                                                      3775
      VCUM(IX)=VCUM(IX)+XX(I)                                           3776
      TRAV(IX)=UCOST(IX,VCUM(IX))                                       3777
   50 CONTINUE                                                          3778
   60 CONTINUE                                                          3780
      X(6)=XXX                                                          3781
*                                                                       3782
*                                                                       3783
  100 DO 120 I=1,NP                                                     3784
      N=LPJ(I)                                                          3790
      SUM=0                                                             3800
      DO 121 J=1,N                                                      3810
```

```
      121 SUM=SUM + TRAV(PATH(J,I))                                      3820
      120 U(I)=SUM                                                       3830
60        *                                                              3840
          *                                                              3850
          DO 140 I=1,NP                                                  3860
      140 DRT(I)=0                                                       3870
          *                                                              3880
65        *                                                              3890
          UBA=1.0E9                                                      3900
          DO 150I= 1,NP                                                  3910
          IF(UBA.LE.U(I))GOTO 150                                        3920
          UBA=U(I)                                                       3930
70        IBA=I                                                          3940
      150 CONTINUE                                                       3950
          *                                                              3960
          SUM=0                                                          3970
          ALPHA=0                                                        3980
75        DO 160 J=1,NP                                                  3990
          IF(J.EQ.IBA) GOTO 160                                          3991
          D=U(J)-UBA                                                     4000
          GAP=2.0*D/(UBA+U(J)+0.001)                                     4010
          IF(GAP.LE.EPSI.OR. X(J).LE.0.1)GOTO 160                        4020
80        DRT(J)=-D*X(J)                                                 4030
          SUM=SUM + DRT(J)                                               4040
          IF(ALPHA.LT.D)ALPHA=D                                          4050
      160 CONTINUE                                                       4060
          *                                                              4070
85        SUM= - SUM                                                     4090
      170 DRT(IBA)= SUM                                                  4100
          *                                                              4110
          IF(ABS(SUM).LE.0.0)GOTO 315                                    4120
          IF(ALPHA.LE.0.0)ALPHA=1.0                                      4130
90    190 DO 200 J=1,NP                                                  4140
      200 DRT(J)=DRT(J)/ALPHA                                            4150
          *                                                              4160
          F=.FALSE.                                                      4170
          DO 205 I=1,NLINK                                               4180
95    205 DELTA(I)=0.0                                                   4190
          *                                                              4200
          DO 210 I=1,NP                                                  4210
          N=LPJ(I)                                                       4220
          D=DRT(I)                                                       4230
100       IF(ABS(D).LE.0.0)GOTO 210                                      4240
          DO 215 J=1,N                                                   4250
      215 DELTA(PATH(J,I))= DELTA(PATH(J,I)) +D                          4260
      210 CONTINUE                                                       4270
          *                                                              4280
105       CALL GSM(LAMBDA,XLIM,ACCUR)                                    4290
          *                                                              4300
          DO 300 I=1,NLINK                                               4310
          IF(ABS(DELTA(I)).LE.0.0)GOTO 300                               4320
          VCUM(I)= VCUM(I) + LAMBDA*DELTA(I)                             4330
110       TRAV(I)=UCOST(I,VCUM(I))                                       4350
      300 CONTINUE                                                       4360
          DO 310 I=1,NP                                                  4370
          X(I)=X(I)+LAMBDA*DRT(I)                                        4371
      310 CONTINUE                                                       4372
```

```
                      GOTO 100                                                  4381
                  315 CONTINUE                                                  4382
                      CALL REVPTH(NADD,EPSI,X(1),UBA)                           4383
                      IF(NADD.GT.0) GOTO 100                                    4396
                *
                *                                                               4400
                *                                                               4410
                  500 CONTINUE                                                  4420
                      LINK(1)=UBA*10000                                         4450
                      LINK(300)=X(6)*10000                                      4450
                      DO 530 I=1,NP                                             4461
                  530 IFLOW(I)=X(I)*10000                                       4462
                  540 VAL=0.                                                    4463
                      DO 600 I=1,NLINK                                          4470
                      VAL=VAL+TCOST(I,VCUM(I))                                  4480
                  600 CONTINUE                                                  4490
                      RETURN                                                    4500
                      END                                                       4520
                                                                                4530
```

CARD NR.	SEVERITY	DETAILS	DIAGNOSIS OF PROBLEM
102	I	NP	THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.
126	I	NP	THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 EQFIX

VARIABLES	SN	TYPE		RELOCATION
503	ACCUR	REAL		
532	D	REAL		
522	DIFF	REAL		
0	EPSI	REAL		F.P.
0	FUPQ	REAL		F.P.
502	GMAX	REAL		
530	IBA	INTEGER		
0	IPAIR	INTEGER	*UNUSED	F.P.
525	J	INTEGER		
0	LINK	INTEGER	ARRAY	/ /
0	MEMO1	INTEGER	ARRAY	/ /
524	N	INTEGER		
374	ND	INTEGER	ARRAY	NET1
1	NLINK	INTEGER		NET1
2	NO	INTEGER		/ /
4	NP	INTEGER		/ /
0	PTHFIL	INTEGER	*UNUSED	F.P.
517	THETA	REAL		
516	TTIN	REAL		
0	UBA	REAL		F.P.
531	ALPHA	REAL		
454	DELTA	REAL	ARRAY	/ /
534	DRT	REAL	ARRAY	
0	F	LOGICAL		F.P.
523	GAP	REAL		
520	I	INTEGER		
5	IFLOW	INTEGER	ARRAY	/ /
526	IX	INTEGER		
515	LAMBDA	REAL		
12	LPJ	INTEGER	ARRAY	/ /
1604	MEMO2	INTEGER	ARRAY	/ /
533	NADD	INTEGER		
3	NDEST	INTEGER		/ /
0	NNODE	INTEGER		NET1
2	NOLAB	INTEGER	ARRAY	NET1
17	PATH	INTEGER	ARRAY	/ /
527	SUM	REAL		
1524	TRAV	REAL	ARRAY	NET1
550	U	REAL	ARRAY	
0	VAL	REAL		F.P.

```
      SUBROUTINE GSM(LAMBDA,XLIM,ACCUR)                                 4540
      COMMON /STAT/RUNSEC(10)                                           4550
      LOGICAL ELAST                                                     4560
      EQUIVALENCE(SHSEC,RUNSEC(3))                                      4570
      REAL LAMBDA                                                       4580
*                                                                       4590
      CALL SECOND(TTIN)                                                 4600
      TAU=(1.0+SQRT(5.))*.5                                             4610
*                                                                       4620
      G=0.0                                                             4630
      D=XLIM                                                            4640
      XB=D/TAU                                                          4650
      XA=D-XB                                                           4660
*                                                                       4670
      FXA=FCTVLU(XA)                                                    4680
      FXB=FCTVLU(XB)                                                    4690
*                                                                       4700
   77 IF(FXA-FXB) 80,80,78                                              4710
   78 G=XA                                                              4720
      XA=XB                                                             4730
      XB=G+D-XA                                                         4740
      FXA=FXB                                                           4750
      FXB=FCTVLU(XB)                                                    4751
      GOTO 82                                                           4761
*                                                                       4762
   80 D=XB                                                              4763
      XB=XA                                                             4764
      XA=G+D-XB                                                         4765
      FXB=FXA                                                           4766
      FXA=FCTVLU(XA)                                                    4767
*                                                                       4768
   82 IF(D-G-ACCUR) 83,83,77                                            4769
*                                                                       4770
   83 LAMBDA=XB                                                         4771
      VAL=FXB                                                           4772
      IF(FXB.LT.FXA) GOTO 1000                                          4773
      VAL=FXA                                                           4774
      LAMBDA=XA                                                         4775
*                                                                       4776
 1000 CALL SECOND (TTOUT)                                               4777
      SHSEC=SHSEC+TTOUT-TTIN                                            4778
      RETURN                                                            4779
      END                                                               4780
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 GSM

VARIABLES	SN	TYPE	RELOCATION				
0	ACCUR	REAL	F.P.	146	D	REAL	
142	ELAST	LOGICAL	*UNDEF	151	FXA	REAL	
152	FXB	REAL		145	G	REAL	

```
      1              REAL FUNCTION FCTVLU(LAMBDA)                                          4781
                     COMMON /NET1/ NNODE,NLINK,NOLAB(250),ND(600),TRAV(600),VCUM(600)      4782
                     COMMON MEMO1(900),MEMO2(550)                                          4783
      5              DIMENSION DELTA(100)                                                  4784
                     REAL LAMBDA                                                           4785
                     EQUIVALENCE(DELTA(1),MEMO1(301))                                      4786
                *                                                                          4787
                     F=0.0                                                                 4788
                     DO 10 I=1,NLINK                                                       4789
     10              VOL=VCUM(I)+(LAMBDA*DELTA(I))                                         4790
                     F=F+TCOST(I,VOL)                                                      4791
                  10 CONTINUE                                                              4792
                     FCTVLU=F                                                              4793
                     RETURN                                                                4794
     15              END                                                                   4795
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
5 FCTVLU

VARIABLES	SN	TYPE		RELOCATION					
454	DELTA	REAL	ARRAY	/ /	45	F	REAL		
44	FCTVLU	REAL			46	I	INTEGER		
0	LAMBDA	REAL		F.P.	0	MEMO1	INTEGER	ARRAY	/ /
1604	MEMO2	INTEGER	ARRAY	/ /	374	ND	INTEGER	ARRAY	NET1
1	NLINK	INTEGER		NET1	0	NNODE	INTEGER		NET1
2	NOLAB	INTEGER	ARRAY	NET1	1524	TRAV	REAL	ARRAY	NET1
2654	VCUM	REAL	ARRAY	NET1	47	VOL	REAL		

EXTERNALS	TYPE	ARGS
TCOST	REAL	2

STATEMENT LABELS
0 10

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
23	10	* I	9 12	138	EXT REFS

COMMON BLOCKS	LENGTH
NET1	2052
/ /	1450

STATISTICS

PROGRAM LENGTH	528	42
CM LABELED COMMON LENGTH	40048	2052
CM BLANK COMMON LENGTH	26528	1450

```
      SUBROUTINE P2(U,S,XCHAP,XCHAPM1,THETA,J,EPSI)                     7510
*                                                                       7520
* RESOLUTION DU PROBLEME DE L ETAPE 2                                   7530
* U CONTIENT LES U OPTIMAUX DES ETAPE 1,...,J-1 (PROB. P1)              7540
* S CONTIENT LES S(J) ("CONSUMER'S SURPLUS")                            7550
* XJ ET XCHAP SONT LA SOLUTION OPTIMALE DU PB. 1 ETAPES 1 ,....,J-1     7560
*                                                                       7570
      COMMON /STAT/RUNSEC(10)                                           7571
      EQUIVALENCE(RUNSEC(5),MPSEC)                                      7572
      DIMENSION U(100),S(100),PHI(100)                                  7580
      REAL MPSEC                                                        7581
      CALL SECOND(TTIN)                                                 7582
      JJ=J-1                                                            7590
      IF(J.EQ.1) GOTO 130                                               7600
      LJ=S(J)+(U(J)*FCTDEM(U(J),THETA))+DEMINT(U(J),THETA)              7601
      DO 1 I=1,JJ                                                       7602
      L=S(I)+(U(I)*FCTDEM(U(J),THETA))+DEMINT(U(J),THETA)               7603
      IF(L.GT.LJ) GOTO 5                                                7604
    1 CONTINUE                                                          7605
      XCHAPM1=XCHAP                                                     7606
      XCHAP=FCTDEM(U(J),THETA)                                          7607
      CALL SECOND(TTOUT)                                                7608
      MPSEC=MPSEC+TTOUT-TTIN                                            7609
      RETURN                                                            7610
    5 DO 10 I=1,JJ                                                      7611
      PHI(I)=(S(J)-S(I))/(U(I)-U(J))                                    7620
   10 CONTINUE                                                          7630
      PHIZJ1=FCTDEM(0.0,THETA)                                          7640
      PHIZJ2=0.0                                                        7650
      PHIXC=XCHAP                                                       7660
*                                                                       7670
      DO 100 I=1,JJ                                                     7680
      IF(PHI(I).LE.PHIXC) GOTO 50                                       7690
      IF(PHIZJ1.LT.PHI(I)) GOTO 100                                     7700
      PHIZJ1=PHI(I)                                                     7720
      GOTO 100                                                          7730
   50 IF(PHIZJ2.GT.PHI(I)) GOTO 100                                     7740
      PHIZJ2=PHI(I)                                                     7760
  100 CONTINUE                                                          7770
*                                                                       7780
*                                                                       7790
      IF(PHIZJ1.LT.PHIZJ2) GOTO 120                                     7800
      T=PHIZJ1                                                          7810
      PHIZJ1=PHIZJ2                                                     7820
      PHIZJ2=T                                                          7830
*                                                                       7870
*                                                                       7880
  120 PHIZB=PHIZJ2                                                      7890
      IF(FCTDEM(U(J),THETA).GT.PHIZJ1) GOTO 150                         7910
      PHIZB=PHIZJ1                                                      7920
      GOTO 150                                                          7933
  130 CONTINUE                                                          7934
      XCHAPM1=XCHAP                                                     7935
      XCHAP=FCTDEM(U(1),THETA)                                          7936
      CALL SECOND(TTOUT)                                                7937
      MPSEC=MPSEC+TTOUT-TTIN                                            7938
      RETURN                                                            7953
```

```
                150 CONTINUE                                                    7980
                    XCHAPM1=XCHAP                                               7981
     60             XCHAP=PHIZB                                                 7990
                    CALL SECOND(TTOUT)                                          7991
                    MPSEC=MPSEC+TTOUT-TTIN                                      7992
                    RETURN                                                      8010
                    END                                                         8020
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 P2

VARIABLES	SN	TYPE		RELOCATION					
0	EPSI	REAL	*UNUSED	F.P.	300	I	INTEGER		
0	J	INTEGER		F.P.	276	JJ	INTEGER		
301	L	INTEGER			277	LJ	INTEGER		
4	MPSEC	REAL		STAT	310	PHI	REAL	ARRAY	
305	PHIXC	REAL			307	PHIZB	REAL		
303	PHIZJ1	REAL			304	PHIZJ2	REAL		
0	RUNSEC	REAL	ARRAY	STAT	0	S	REAL	ARRAY	F.P.
306	T	REAL			0	THETA	REAL		F.P.
275	TTIN	REAL			302	TTOUT	REAL		
0	U	REAL	ARRAY	F.P.	0	XCHAP	REAL		F.P.
0	XCHAPM1	REAL		F.P.					

EXTERNALS	TYPE	ARGS			
DEMINT	REAL	2	FCTDEM	REAL	2
SECOND		1			

STATEMENT LABELS

0	1	121	5	0	10		
151	50	170	100	205	120		
223	130	244	150				

LOOPS	LABEL		INDEX	FROM-TO	LENGTH	PROPERTIES	EXT REFS	EXITS
52	1	*	I	16 19	258			
122	10		I	25 27	118	OPT		
146	100		I	32 39	258	OPT		

COMMON BLOCKS	LENGTH
STAT	10

STATISTICS

PROGRAM LENGTH	5038	323
CM LABELED COMMON LENGTH	128	10

```
      SUBROUTINE GB(IPAIR,PTHFIL,KOPT,EPSI)                              8030
     *                                                                   8040
     * METHODE DE BENDERS GENERALISEE                                    8050
     *                                                                   8060
      COMMON MEMO1(900),MEMO2(550)                                       8070
      INTEGER PTHFIL                                                     8080
      LOGICAL FLAG,F                                                     8090
      DIMENSION LINK(300),U(100),S(100)                                  8100
      EQUIVALENCE(LINK(1),MEMO1(1)),(NP,LINK(5))                         8110
     *                                                                   8120
      J=1                                                                8130
      BSUP=1.0E11                                                        8140
      XJ=0.0                                                             8150
     *                                                                   8160
      KOPT=KOPT+1                                                        8170
      FLAG=.TRUE.                                                        8180
      CALL SECOND (TTIN)                                                 8190
      CALL READMS(PTHFIL,LINK,300,IPAIR)                                 8200
      IF(NP.EQ.0) RETURN                                                 8210
      UPQ=LINK(1)/10000.                                                 8220
      THETA=LINK(2)/10000.                                               8230
      FUPQ=FCTDEM(UPQ,THETA)                                             8231
     *                                                                   8240
     *                                                                   8250
    1 CALL EQFIX(EPSI,FUPQ,UBA,VAL,F,IPAIR,PTHFIL)                       8260
      U(J)=UBA                                                           8280
      FUBA=FCTDEM(UBA,THETA)                                             8290
     *                                                                   8320
      GAP=2.0*ABS(FUPQ-FUBA)/(FUPQ+FUBA+0.001)                           8321
      IF(GAP.LT.0.1) GOTO 1000                                           8322
      FLAG=.FALSE.                                                       8330
      S(J)=VAL-(UBA*FUPQ)                                                8340
      CALL P2(U,S,FUPQ,FUPQM1,THETA,J,EPSI)                              8350
      GAP=2.0*ABS(FUPQ-FUPQM1)/(FUPQ+FUPQM1+0.001)                       8355
      IF(GAP.LT.0.1) GOTO 1000                                           8356
      J=J+1                                                              8360
      IF(J.GT.100) CALL EXIT                                             8361
      GOTO 1                                                             8370
 1000 IF(.NOT.(FLAG.AND.F)) KOPT=1                                       8380
      CALL WRITMS(PTHFIL,LINK,300,IPAIR,1,1)                             8390
      RETURN                                                             8400
      END                                                                8410
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS

4	GB

VARIABLES	SN	TYPE	RELOCATION				
215	BSUP	REAL		0	EPSI	REAL	F.P.
213	F	LOGICAL		212	FLAG	LOGICAL	
225	FUBA	REAL		222	FUPQ	REAL	
227	FUPQM1	REAL		226	GAP	REAL	

A2.6 Sous-routines et fonctions communes à tous les programmes précédents

```
      1                SUBROUTINE RDOD(INFIL,NCENT,OUTFIL,INDEX)                    2590
                       DIMENSION ROW(20),INDEX(20)                                  2600
                       INTEGER OUTFIL                                               2610
                     *                                                              2620
      5               4000 FORMAT(15F7.4)                                          2630
                      4001 FORMAT(8F10.4)                                           2640
                     *                                                              2650
                       REWIND INFIL                                                 2660
                       REWIND OUTFIL                                                2670
     10                DO 10 I=1,NCENT                                             2680
                    10 INDEX(I)=-I                                                  2690
                       DO 20 I=1,NCENT                                              2700
                       READ (INFIL,4001) (ROW(J),J=1,NCENT)                         2710
                       SUM= 0                                                       2720
     15                DO 30 J=1,NCENT                                             2730
                    30 SUM=SUM+ROW(J)                                               2740
                       IF(SUM.LE.0.0)GOTO 20                                        2750
                       INDEX(I)=I                                                   2760
                       WRITE(OUTFIL,4000) (ROW(J),J=1,NCENT)                        2770
     20                PRINT 4001,(ROW(J),J=1,NCENT)                               2771
                    20 CONTINUE                                                     2780
                       RETURN                                                       2790
                       END                                                          2800
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 RDOD

VARIABLES	SN	TYPE		RELOCATION					
123	I	INTEGER			0	INDEX	INTEGER	ARRAY	F.P.
0	INFIL	INTEGER		F.P.	124	J	INTEGER		
0	NCENT	INTEGER		F.P.	0	OUTFIL	INTEGER		F.P.
126	ROW	REAL	ARRAY		125	SUM	REAL		

FILE NAMES	MODE
OUTPUT	FMT

STATEMENT LABELS

0	10			73	20			0	30
100	4000	FMT		102	4001	FMT			

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
24	10	I	10 11	5B	INSTACK
33	20	* I	12 21	43B	EXT REFS NOT INNER
45	30	J	15 16	5B	INSTACK

STATISTICS
PROGRAM LENGTH 1708 120

```
      SUBROUTINE RDNET(INNET,NCENT)                                      2810
C                                                                        2820
C        CURNET CONTAINS THE UNIT NUMBER OF THE FILE FROM WHICH          2830
C             THE LINK DATA WILL BE READ.                                2840
C                                                                        2850
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)    2850
      COMMON /LDATA/TO(400),BETA(400),LTYP(400)                          2870
*                                                                        2880
      DATA MAXERR/3/                                                     2890
      DATA MAXNOD/250/                                                   2900
 1106 FORMAT (1H0,20X,"DIMENSIONS SPECIFIEES TROUVEES"/                  2910
     +               23X,   "- SOMM",I9,I10/                            2920
     +               23X,   "- LIEN",I9,I10/                            2930
     +               23X,   "- CENT",I9/ 1H1)                           2940
 2001 FORMAT(1H0,20X,"0601E LIENS NON TRIES. LIEN :",I4//)               2941
C                                                                        2950
C--------READ DIMENSION RECORD                                           2960
      REWIND INNET                                                       2970
      READ(INNET,3000) NCENT,NNODE,NLINK                                 2980
 3000 FORMAT(5I5)                                                        2990
 3001 FORMAT(2I5,2F12.6,I5)                                              2991
C                                                                        3000
   35 DO 50 I=1,MAXNOD                                                   3010
   50     NOLAB(I)=0                                                    3020
      NOLAB(1)=1                                                         3030
      I=0                                                                3040
      MNO=0                                                              3050
      IERR = 0                                                           3060
      J=1                                                                3070
   55 I=I+1                                                              3080
C--------READ LINK RECORD                                               3090
      READ( INNET, 3001) NO,ND(I),TO(I),BETA(I),LTYP(I)                  3100
      IF (EOF(INNET)) 59,56,59                                           3110
   56 MNO=MAXO (NO,ND(I),MNO)                                            3120
C--------SET UP LINK ORIGIN INDEX TABLE NOLAB                            3130
      IF(NO-J)57,55,58                                                   3140
   57     PRINT 2001,I                                                  3150
          CALL EXIT                                                      3160
   58 NOLAB(NO) = I                                                      3170
      J=NO                                                               3180
      GOTO 55                                                            3190
C                                                                        3200
   59 I=I-1                                                              3210
  110 PRINT 1106, NNODE,MNO, NLINK,I, NCENT                              3250
      NLINK=I                                                            3260
      NNODE = MNO                                                        3270
      NACTIV=0                                                           3280
      NOLAB(NNODE+1) = NLINK + 1                                         3290
      RETURN                                                             3300
      END                                                                3310
```

SYMBOLIC REFERENCE MAP (R=1)

```
      SUBROUTINE LOADTR (NN,PTHFIL)                                          3330
*                                                                            3340
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)        3350
      COMMON /OD1/ NCENT, NPAIR, ROW(20), KORIG(20),NOPR(20)                 3360
      COMMON MEMO1(800),MEMO2(550)                                           3370
      DIMENSION LINK(300),PATH(50,5),IFLOW(5),LPJ(5)                         3380
      DIMENSION LPR(250),NPR(250),VK(20)                                     3390
      INTEGER PTHFIL,PATH                                                    3400
      EQUIVALENCE (LPR(1),MEMO2(1)),(NPR(1),MEMO2(251)),                     3420
     +            (VK(1),MEMO2(501))                                         3430
      EQUIVALENCE (LINK(1),MEMO1(1)),(PATH(1,1),LINK(16)),                   3440
     + (NO,LINK(3)),(NDEST,LINK(4)),(NP,LINK(5)),                            3450
     +       (IFLOW(1),LINK(5)),(LPJ(1),LINK(11))                            3460
*                                                                            3470
C                                                                            3480
      CALL MINTRE(NN,LPR,VK,NPR)                                             3490
      LPR(NN)=0                                                              3500
C                                                                            3510
      DO 30 I=1,300                                                          3520
   30 LINK(I)=0                                                              3530
      NTOD=0                                                                 3540
      DO 50 I=1,NCENT                                                        3550
      UPQ=50.0                                                               3560
      IF(ROW(I).LE.0.0) GOTO 50                                              3570
      TRIP=FCTDEM(UPQ,ROW(I))                                                3580
      IF(UPQ.LT.VK(I)) GOTO 50                                               3590
   41 NEXT=I                                                                 3600
      NTOD=NTOD+1                                                            3610
      NUM=0                                                                  3620
   40 L=LPR(NEXT)                                                            3630
      VCUM(L)=VCUM(L)+TRIP                                                   3640
      NUM=NUM+1                                                              3650
      PATH(NUM,I)=L                                                          3660
      NEXT=NPR(NEXT)                                                         3670
      IF(NEXT.NE.NN)GOTO 40                                                  3680
      IFLOW(I)=TRIP*10000.                                                   3690
       LPJ(I)=NUM                                                            3700
      LINK(1)=UPQ*10000                                                      3710
      LINK(2)=ROW(I)*10000                                                   3720
      NO=NN                                                                  3730
      NDEST=I                                                                3740
      NP=1                                                                   3750
      NPAIR=NPAIR+1                                                          3760
      CALL WRITMS(PTHFIL,LINK,300,NPAIR,0,1)                                 3770
   50 CONTINUE                                                               3780
      NOPR(NN)=NTOD                                                          3790
      RETURN                                                                 3800
      END                                                                    3810
```

SYMBOLIC REFERENCE MAP (R=1)

```
      SUBROUTINE REVPTH(NN,K1,K2,PTHFIL,NADD,NDEL,EPSI)                 5410
*                                                                       5420
*------REVISE THE SET OF CURRENT PATHS FOR OD PAIR                      5430
*------K1 TO K2.DELETE PATH WITH LESS THAN 1 UNIT OF FLOW,             5440
*------ADD NEW SHORTEST PATH.                                          5450
*                                                                       5460
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)   5470
      COMMON MEMO1(800),MEMO2(550)                                      5480
      DIMENSION LINK(300),PATH(50,5),IFLOW(5),LPJ(5)                    5490
      DIMENSION LPR(250),NPR(250),VK(20)                                5500
      INTEGER PTHFIL,PATH                                               5510
      EQUIVALENCE (LPR(1),MEMO2(1)),(NPR(1),MEMO2(251)),                5520
     +                  (VK(1),MEMO2(501))                              5530
      EQUIVALENCE (LINK(1),MEMO1(1)),(PATH(1,1),LINK(16)),              5540
     +      (NDEST,LINK(4)),(NP,LINK(5)),                               5550
     +       (IFLOW(1),LINK(6)),(LPJ(1),LINK(11))                       5560
      DATA EPS2/0.1/                                                    5561
*                                                                       5570
*                                                                       5580
      DO 100 JJ= K1,K2                                                  5590
      CALL READMS (PTHFIL,LINK,300,JJ)                                  5600
      NOLD=NP                                                           5610
      NEWNP=0                                                           5620
C                                                                       5630
      DEM=LINK(1)                                                       5640
      IF(DEM.GT.0.0)GOTO 530                                            5650
      IF(NP.EQ.0)100,560                                                5660
  530 DO 550 I=1,NP                                                     5670
      IF(IFLOW(I).LE.0)GOTO 550                                         5680
      NEWNP=NEWNP+1                                                     5690
      IF(I.EQ.NEWNP)GOTO 550                                            5700
      IFLOW(NEWNP)=IFLOW(I)                                             5710
      LPJ(NEWNP)=NUM=LPJ(I)                                             5720
      LPJ(I)=0                                                          5730
      IFLOW(I)=0                                                        5740
      DO 540 J=1,NUM                                                    5750
  540 PATH(J,NEWNP)=PATH(J,I)                                           5760
  550 CONTINUE                                                          5770
  560 NP=NEWNP                                                          5780
      CALL WRITMS (PTHFIL,LINK,300,JJ,1,1)                              5790
      NDEL=NDEL +NOLD-NEWNP                                             5800
  100 CONTINUE                                                          5810
*                                                                       5820
*------ADD NEW SHORTEST PATH                                           5830
*                                                                       5840
      CALL  MINTRE(NN,LPR,VK,NPR)                                       5850
      LPR(NN)=0                                                         5860
C                                                                       5870
      DO 50 J=K1,K2                                                     5880
   50 CALL READMS(PTHFIL,LINK,300,J)                                    5890
      IF(NP.EQ.5)GOTO 50                                                5900
      UPQ= LINK(1)/10000.0                                              5910
      R= (UPQ-VK(NDEST))/(VK(NDEST) + 0.001)                            5920
      IF(R.LE.EPS2)GOTO 50                                              5930
      NP=NP+1                                                           5940
      IFLOW(NP)=0                                                       5950
      NUM=0                                                             5960
```

```
                      NADD=NADD +1                                              5970
                      NEXT=NDEST                                                5980
      60           40 L=LPR(NEXT)                                               5990
                      NUM=NUM+1                                                 6000
                      PATH(NUM,NP)=L                                            6010
                      NEXT=NFR(NEXT)                                            6020
                      IF(NEXT.NE.NN)GOTO 40                                     6030
      65              LPJ(NF)=NUM                                               6040
                      CALL WRITMS(PTHFIL,LINK,300,J,1,1)                        6050
                   50 CONTINUE                                                  6060
                      RETURN                                                    6070
                      END                                                       6080
```

CARD NR.	SEVERITY	DETAILS	DIAGNOSIS OF PROBLEM
32	I	NP	THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.
33	I	NP	THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.
34	I	NP	THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.
35	I	NP	THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.
37	I	NP	THIS STATEMENT MAY REDEFINE A CURRENT LOOP CONTROL VARIABLE OR PARAMETER, OPTIMIZATION INHIBITED.

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 REVPTH

VARIABLES	SN	TYPE	RELOCATION						
256	DEM	REAL			0	EPSI	REAL	*UNUSED	F.P.
245	EPS2	REAL			257	I	INTEGER		
5	IFLOW	INTEGER	ARRAY	/ /	261	J	INTEGER		
253	JJ	INTEGER			0	K1	INTEGER		F.P.
0	K2	INTEGER		F.P.	265	L	INTEGER		
0	LINK	INTEGER	ARRAY	/ /	12	LPJ	INTEGER	ARRAY	/ /
1440	LPR	INTEGER	ARRAY	/ /	0	MEMO1	INTEGER	ARRAY	/ /
1440	MEMO2	INTEGER	ARRAY	/ /	0	NADD	INTEGER		F.P.
374	ND	INTEGER	ARRAY	NET1	0	NDEL	INTEGER		F.P.
3	NDEST	INTEGER		/ /	255	NEWNP	INTEGER		
264	NEXT	INTEGER			1	NLINK	INTEGER		NET1
0	NN	INTEGER		F.P.	0	NNODE	INTEGER		NET1
2	NOLAB	INTEGER	ARRAY	NET1	254	NOLD	INTEGER		
4	NP	INTEGER		/ /	2032	NPR	INTEGER	ARRAY	/ /
260	NUM	INTEGER			17	PATH	INTEGER	ARRAY	/ /
0	PTHFIL	INTEGER		F.P.	263	R	REAL		
1214	TRAV	REAL	ARRAY	NET1	262	UPQ	REAL		
2034	VCUM	REAL	ARRAY	NET1	2424	VK	REAL	ARRAY	/ /

EXTERNALS	TYPE	ARGS			
MINTRE		4	READMS		4
WRITMS		6			

```
      SUBROUTINE  MINTRE(NN,LPR,VK,NPR)                                 6290
C                                                                       6300
C        TRAFFIC - 2     SUBPROGRAM  67                                 6310
C                                                                       6320
C---------CONSTRUCT SHORTEST PATH TO ALL DESTINATIONS FORM ORIGIN NN.   6330
C                                                                       6340
C     LPR(N)  WILL CONTAIN THE NUMBER OF THE LINK INCIDENT              6350
C             TO NODE I, BELONGING TO A SHORTEST PATH ORIGINATING       6360
C             AT ORIGIN NN (PERMANENT LABEL OF NODE).                   6370
C     LAB(K)  CONTAINS THE CURRENT ADDRESS OF A NODE K IF               6380
C             IT IN THE HEAP OF CANDIDATES SPECIFIED IN NSET, TSET.     6390
C     NSET AND TSET CONTAIN THE VALUES ASSOCIATED WITH NODE (K)         6400
C             IN BINARY HEAP ORDER)                                     6410
C     NSET(I) CONTAINS THE NUMBER OF THE LINK INCIDENT TO NODE (K))     6420
C     TSET CONTAINS THE PATH TIME TO THE NODE (K))                      6430
C                                                                       6440
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)   6450
      COMMON /OD1/ NCENT, NPAIR, ROW(20), KORIG(20),NOPR(20)            6460
      COMMON MEMO1(800),MEMO2(550)                                      6470
      COMMON/STAT/RUNSEC(10)                                            6480
      DIMENSION LPR(250),NPR(250),VK(20)                                6490
      DIMENSION TSUM(250),LAB(250),TSET(100),NSET(100),MSET(100)        6500
      EQUIVALENCE (TSUM(1),MEMO1(1)),(LAB(1),MEMO1(251)),               6510
     +(TSET(1),MEMO1(501)),(NSET(1),MEMO1(601)),(MSET(1),MEMO1(701))    6520
      EQUIVALENCE( PTSEC,RUNSEC(4))                                     6530
C                                                                       6540
C                                                                       6550
  100 FORMAT (1H0,20X,"6700E CAPACITE DE NSET DEPASSEE. ",              6560
     +               "ORIGINE ACTUELLE :",I5//)                         6570
C                                                                       6580
C      MIN PATH USING DIJKSTRA ALGORITHM AND HEAPS STRUCTURE            6590
      CALL SECOND(TTIN)                                                 6600
C                                                                       6610
      N=NN                                                              6620
      NA=1                                                              6630
      NB=0                                                              6640
C                                                                       6650
C     INITIALISE TSUM AND LAB                                           6660
      DO 2 I=1,NNODE                                                    6670
      TSUM(I)=1.0E10                                                    6680
      LAB(I)=0                                                          6690
      LPR(I)=0                                                          6700
    2 CONTINUE                                                          6710
C                                                                       6720
      TSUM(N)=0                                                         6730
      LPR(N)=999999                                                     6740
C                                                                       6750
C     LABEL NEW NODES                                                   6760
C                                                                       6770
    6 J=NOLAB(N)                                                        6780
C                                                                       6790
      IF(J.EQ.0)GOTO 13                                                 6800
      N1 = N                                                            6810
    4 N1 = N1 + 1                                                       6820
      JE = NOLAB(N1) - 1                                                6830
      IF(JE.LT.0) 4,5                                                   6840
C                                                                       6850
```

```
                5 DO 7 I= J,JE                                                  6850
                8 K=ND(I)                                                       6860
60                IF(TSUM(K).LE.0.9E10) GO TO 7                                 6870
                  VAL=TSUM(N)+TRAV(I)                                           6880
                  IF(LAB(K).NE.0) GO TO 11                                      6890
               10     LR=NB=NB+1                                                6900
                      IF(NB.LE.200)GOTO 33                                      6910
65                        PRINT 100,NN                                          6920
                          CALL EXIT                                             6930
               33     LAB(K)=NB                                                 6940
                      GOTO 12                                                   6950
               11 LR=LAB(K)                                                     6960
70                IF(VAL.GE.TSET(LR))GOTO 7                                     6970
               12 TSET(LR)=VAL                                                  6980
                  NSET(LR)=I                                                    6990
                  MSET(LR) = N                                                  7000
                  JJ=LR                                                         7010
75          C                                                                   7020
            C     SIFT UP                                                       7030
            C                                                                   7040
               20 II=JJ                                                         7050
                  IF(II.EQ.1)GOTO 22                                            7060
80                JJ=II/2                                                       7070
                  IF(VAL.GE.TSET(JJ)) GO TO 22                                  7080
               21     TSET(II)=TSET(JJ)                                         7090
                      KK=NSET(II)=NSET(JJ)                                      7100
                      MSET(II) = MSET(JJ)                                       7110
85                    KK=ND(KK)                                                 7120
                      LAB(KK)=II                                                7130
                      GOTO 20                                                   7140
               22 TSET(II)=VAL                                                  7150
                  NSET(II)=I                                                    7160
90                MSET(II) = N                                                  7170
                  LAB(K)=II                                                     7180
                7 CONTINUE                                                      7190
            C                                                                   7200
            C                                                                   7210
95          C     PERMANENT LABEL                                               7220
            C                                                                   7230
               13 CONTINUE                                                      7240
                  IF(NB.LE.0) GO TO 40                                          7250
               42  CONTINUE                                                    7260
100               I=NSET(1)                                                     7270
                  N=ND(I)                                                       7280
                  TSUM(N)=TSET(1)                                               7290
                  LPR(N)=I                                                      7300
                  NPR(N) = MSET(1)                                              7310
105               NA=NA+1                                                       7320
                  IF(NA.LT.NNODE)GOTO 45                                        7330
               40     CALL SECOND(TTOUT)                                        7340
                      PTHSEC = PTHSEC + TTOUT -TTIN                             7350
                      DO 44 I=1,NCENT                                           7360
110            44     VR(I) = TSUM(I)                                           7370
                      RETURN                                                    7380
            C                                                                   7390
            C     DELETE THE LAST NODE LABELED                                  7400
            C                                                                   7410
                                                                                7420
```

```
      45 CONTINUE
         VAL = TSET(1)=TSET(NB)
         I=NSET(1)=NSET(NB)
         NI = MSET(1) = MSET(NB)
         NB=NB-1
         K=ND(I)
         LAB(K)=1
         JJ=1
C
C        PUSH DOWN
C
      14 II=JJ
         JJ=II*2
         IF(JJ-NB)16,17,18
      16 IF(TSET(JJ) .GT.TSET(JJ+1))JJ=JJ+1
      17 IF(VAL.LE.TSET(JJ))GOTO 18
              TSET(II)=TSET(JJ)
              KK=NSET(II)=NSET(JJ)
              MSET(II) = MSET(JJ)
              KK=ND(KK)
              LAB(KK)=II
              GOTO 14
      18 TSET(II)=VAL
         NSET(II)=I
         MSET(II) = NI
         LAB(K)=II
C---------NO PATH IS ALLOWED TO PASS THROUGH A CENTROID (ORIGIN OR
C            DESTINATION)
         IF(N.LE.NCENT) GO TO 13
              GO TO 6
         END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 MINTRE

VARIABLES

SN	NAME	TYPE		RELOCATION	SN	NAME	TYPE		RELOCATION
417	I	INTEGER			427	II	INTEGER		
420	J	INTEGER			422	JE	INTEGER		
426	JJ	INTEGER			423	K	INTEGER		
430	KK	INTEGER			25	KORIG	INTEGER	ARRAY	OD1
372	LAB	INTEGER	ARRAY	/ /	0	LPR	INTEGER	ARRAY	F.P.
425	LR	INTEGER			0	MEMO1	INTEGER	ARRAY	/ /
1440	MEMO2	INTEGER	ARRAY	/ /	1274	MSET	INTEGER	ARRAY	/ /
414	N	INTEGER			415	NA	INTEGER		
416	NB	INTEGER			0	NCENT	INTEGER		OD1
374	ND	INTEGER	ARRAY	NET1	432	NI	INTEGER		
1	NLINK	INTEGER		NET1	0	NN	INTEGER		F.P.
0	NNODE	INTEGER		NET1	2	NOLAB	INTEGER	ARRAY	NET1
52	NOFR	INTEGER	ARRAY	OD1	1	NPAIR	INTEGER		OD1
0	NPR	INTEGER	ARRAY	F.P.	1130	NSET	INTEGER	ARRAY	/ /
421	NI	INTEGER			3	PTHSEC	REAL		STAT

```
      SUBROUTINE OUTPTH(NPAIR,PTHFIL)                                   7740
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)   7750
      COMMON MEMO1(600),MEMO2(550)                                      7760
      DIMENSION LINK(300),PATH(50,5),IFLOW(5),LPJ(5)                    7770
      INTEGER PTHFIL,PATH                                               7780
      EQUIVALENCE (LINK(1),MEMO1(1)),(PATH(1,1),LINK(16)),              7790
     + (NO,LINK(3)),(NDEST,LINK(4)),(NP,LINK(5)),                       7800
     +       (IFLOW(1),LINK(6)),(LPJ(1),LINK(11))                       7810
C                                                                       7820
      PRINT 1000                                                        7830
 1000 FORMAT(1H1,5X,* C H E M I N S  U T I L I S E S*//)                 7840
      NEWO=0                                                            7850
      DO 10 J=1,NPAIR                                                   7860
      CALL READNS(PTHFIL,LINK,300,J)                                    7870
      IF(NO.EQ.NEWO)GOTO 20                                             7880
      NEWO=NO                                                           7890
      PRINT 2000,NEWO                                                   7900
 2000 FORMAT(//5X,* ORIGINE NUMERO :*,I6,/)                             7910
   20 UPQ=LINK(1)/10000.                                                7911
      ACC=FCTDEM(UPQ,(LINK(2)/10000.))                                  7912
C     PRINT 3000,NDEST,ACC,UPQ,NP                                       7913
 3000 FORMAT(2X,*DEST.*,I5,2X,*DEM.*,F10.3,4X,2X,*ACCES.*,F10.3,4X,     7930
     +2X,*N. CHEMINS*,I5/)                                              7940
      DO 30 I=1,NP                                                      7950
      N=LPJ(I)                                                          7960
      PRINT 4000,I,IFLOW(I),(PATH(K,I),K=1,N)                           7970
 4000 FORMAT(2X,I4,* : *,I8,4X,(10I4))                                  7980
   30 CONTINUE                                                          7990
   10 CONTINUE                                                          8000
      RETURN                                                            8010
      END                                                               8020
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 OUTPTH

VARIABLES	SN	TYPE		RELOCATION					
171	ACC	REAL			172	I	INTEGER		
5	IFLOW	INTEGER	ARRAY	/ /	167	J	INTEGER		
174	K	INTEGER			0	LINK	INTEGER	ARRAY	/ /
12	LPJ	INTEGER	ARRAY	/ /	0	MEMO1	INTEGER	ARRAY	/ /
1440	MEMO2	INTEGER	ARRAY	/ /	173	N	INTEGER		
374	ND	INTEGER	ARRAY	NET1	3	NDEST	INTEGER		/ /
166	NEWO	INTEGER			1	NLINK	INTEGER		NET1
0	NNODE	INTEGER		NET1	2	NO	INTEGER		/ /
2	NOLAB	INTEGER	ARRAY	NET1	4	NP	INTEGER		/ /
0	NPAIR	INTEGER		F.P.	17	PATH	INTEGER	ARRAY	/ /
0	PTHFIL	INTEGER		F.P.	1214	TRAV	REAL	ARRAY	NET1
170	UPQ	REAL			2034	VCUM	REAL	ARRAY	NET1

```
      1              REAL FUNCTION UCOST(L,VOL)                                  5240
                     COMMON/LDATA/ TO(400),BETA(400),LTYP(400)                   5250
                *
                                                                                 5260
      5              UCOST=TO(L)+(BETA(L)*VOL)                                   5270
                     RETURN                                                      5280
                     END                                                         5290
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
5 UCOST

VARIABLES	SN	TYPE		RELOCATION					
620 BETA		REAL	ARRAY	LDATA	0	L	INTEGER		F.P.
1440 LTYP		INTEGER	ARRAY	LDATA	0	TO	REAL	ARRAY	LDATA
15 UCOST		REAL			0	VOL	REAL		F.P.

COMMON BLOCKS	LENGTH
LDATA	1200

STATISTICS

PROGRAM LENGTH	168	14
CM LABELED COMMON LENGTH	22608	1200

```
FUNCTION TCOST       73/74   OPT=0 TRACE                     FTN 4.6+428        07/04/78  12.46.14        PAGE      1

                      REAL FUNCTION TCOST(L,VOL)
                      COMMON/LDATA/ TO(400),BETA(400),LTYP(400)
             *
                      TCOST= VOL*(TO(L)+BETA(L)*VOL/2.0)
                      RETURN
                      END


      SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    5  TCOST

VARIABLES       SN  TYPE           RELOCATION
  620  BETA         REAL       ARRAY    LDATA             0  L          INTEGER             F.P.
 1440  LTYP         INTEGER    ARRAY    LDATA            17  TCOST      REAL
    0  TO           REAL       ARRAY    LDATA             0  VOL        REAL                F.P.

COMMON BLOCKS    LENGTH
       LDATA       1200

STATISTICS
  PROGRAM LENGTH                 20B      16
  CM LABELED COMMON LENGTH     2260B    1200
```

```
      1              REAL FUNCTION FCTDEM(U,THETA)                          10
                     DATA GMAX/1000./                                       20
                     FCTDEM=GMAX*EXP(-THETA*U)                              30
                     RETURN                                                 40
      5              END                                                    50
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
5 FCTDEM

VARIABLES	SN	TYPE	RELOCATION				
23	FCTDEM	REAL		22	GMAX	REAL	
0	THETA	REAL	F.P.	0	U	REAL	F.P.

EXTERNALS	TYPE	ARGS
EXP	REAL	1 LIBRARY

STATISTICS
PROGRAM LENGTH 24B 20

```
      1              REAL FUNCTION FPRIM(U,THETA)                                  50
                     DATA GMAX/1000./                                              70
                     FPRIM=-1.0*GMAX*THETA*EXP(-THETA*U)                           80
                     RETURN                                                        90
      5              END                                                          100
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
5 FPRIM

VARIABLES	SN	TYPE	RELOCATION				
26 FPRIM		REAL		24	GMAX	REAL	
0 THETA		REAL	F.P.	0	U	REAL	F.P.

EXTERNALS	TYPE	ARGS
EXP	REAL	1 LIBRARY

STATISTICS
PROGRAM LENGTH 278 23

```
      1              REAL FUNCTION DEMINT(U,THETA)                                  110
                     DATA GMAX/1000./                                               120
                     DEMINT=-EXP(-THETA*U)*GMAX*((1.0/THETA)+U)                     122
                     RETURN                                                         140
      5              END                                                            150
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
5 DEMINT

VARIABLES	SN	TYPE	RELOCATION				
27 DEMINT		REAL		25	GMAX	REAL	
0 THETA		REAL	F.P.	0	U	REAL	F.P.

EXTERNALS	TYPE	ARGS
EXP	REAL	1 LIBRARY

STATISTICS

PROGRAM LENGTH		308	24
SYSLIBN TESTNE2	78081		
SYSLIBN TESTOD5	78081		

A2.7. Programme LIN (algorithme LIN1, section 2.2.2.)

```
      PROGRAM LIN(INPUT,OUTPUT,TAPE1)                                    80
      DIMENSION X(100),A(100),B(100)                                     90
      DATA EPSI/1.0/
*                                                                        100
      READ 9999,DEB,FIN,DIFFMAX,THETA                                    110
 9999 FORMAT(F10.4)                                                      120
      READ 8888,NBSEG                                                    130
 8888 FORMAT(I5)                                                         140
      PRINT 9999,DEB,FIN,DIFFMAX,THETA                                   150
      PRINT 8888,NBSEG                                                   151
*                                                                        152
      IISEG=NBSEG+1                                                      160
      X(1)=DEB                                                           161
      G=0.0                                                              170
      D=DIFFMAX                                                          180
*                                                                        190
   10 DIFF=(G+D)/2.0                                                     200
      ISEG=1                                                             210
*                                                                        220
   20 ISEG=ISEG+1                                                        230
      CALL SEG(DIFF,X,A,B,THETA,ISEG,FIN)                                240
      IF(ISEG.EQ.IISEG.OR.X(ISEG).GT.FIN) GOTO 1000                      250
      GOTO 20                                                            280
*                                                                        290
 1000 IF(ISEG.EQ.IISEG.AND.ABS(X(ISEG)-FIN).LT.EPSI) GOTO 2000           300
      IF(X(ISEG).LT.FIN) G=DIFF                                          310
      IF(X(ISEG).GE.FIN) D=DIFF                                          320
      IF((D-G).LE.1.0E-5) CALL EXIT                                      330
      GOTO 10                                                            340
*                                                                        360
 2000 WRITE(1,8990) NBSEG                                                370
      DO 222 I=1,IISEG                                                   380
      X(I)=F(X(I),THETA)                                                 381
  222 CONTINUE                                                           382
      WRITE(1,9900) (X(I),I=1,IISEG)                                     383
      WRITE(1,9900) (A(I),I=1,NBSEG)                                     390
      WRITE(1,9900) (B(I),I=1,NBSEG)                                     410
 8990 FORMAT(I10)                                                        420
 9900 FORMAT(5E20.12)                                                    430
      PRINT 7777,DIFF                                                    440
 7777 FORMAT(* DIFF. MAXIMALE = *,E15.7)                                 441
      STOP                                                               442
      END                                                                450
                                                                         460
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
6157 LIN

VARIABLES	SN	TYPE	RELOCATION				
6604 A		REAL	ARRAY	6750 B	REAL	ARRAY	
6434 D		REAL		6425 DEB	REAL		
6435 DIFF		REAL		6427 DIFFMAX	REAL		

```
      SUBROUTINE SEG(DIFF,X,A,B,THETA,ISEG,FIN)                          470
*                                                                        480
*TROUVE LE PT. TERMINAL DU PROCHAIN SEGMENT                              490
*                                                                        500
      DIMENSION X(100),A(100),B(100)                                     510
      DATA EPSI/1.0E-2/                                                  520
*                                                                        530
*                                                                        540
      G=XINIT=X(ISEG-1)                                                  550
      D=FIN*1.5                                                          560
*                                                                        570
   10 PT=(G+D)/2.0                                                       580
      PENTE=(PT-XINIT)/(F(PT,THETA)-F(XINIT,THETA))                      590
      ORD=-1.0*F(PT,THETA)*PENTE+PT                                      600
*                                                                        620
      CALL GSM(PTMAX,XMAX,PENTE,ORD,XINIT,PT,THETA)                      630
*                                                                        640
      IF(ABS(XMAX-DIFF).LT.EPSI) GOTO 1000                               650
      IF(XMAX.GT.DIFF) D=PT                                              660
      IF(XMAX.LT.DIFF) G=PT                                              670
      IF((D-G).LT.EPSI) GOTO 1000                                        671
      GOTO 10                                                            680
 1000 X(ISEG)=PT                                                         690
      A(ISEG-1)=PENTE                                                    700
      B(ISEG-1)=ORD                                                      710
      RETURN                                                             720
      END                                                                730
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 SEG

VARIABLES	SN	TYPE	RELOCATION						
0	A	REAL	ARRAY	F.P.	0	B	REAL	ARRAY	F.P.
132	D	REAL			0	DIFF	REAL		F.P.
123	EPSI	REAL			0	FIN	REAL		F.P.
130	G	REAL			0	ISEG	INTEGER		F.P.
135	ORD	REAL			134	PENTE	REAL		
133	PT	REAL			136	PTMAX	REAL		
0	THETA	REAL		F.P.	0	X	REAL	ARRAY	F.P.
131	XINIT	REAL			137	XMAX	REAL		

EXTERNALS	TYPE	ARGS			
F	REAL	2		GSM	7

INLINE FUNCTIONS	TYPE	ARGS	
ABS	REAL	1	INTRIN

STATEMENT LABELS
16 I0 73 1000

```
      SUBROUTINE GSM(PTMAX,XMAX,PENTE,ORD,XINIT,PT,THETA)                740
      DATA ACCUR/1.0E-4/                                                 750
     *                                                                   760
      TAU=(1.0+SQRT(5.))*.5                                              770
     *                                                                   780
      G=XINIT                                                            790
      D=PT                                                               800
      XB=G+((D-G)/TAU)                                                   810
      XA=G+D-XB                                                          820
      YA=F(XA,THETA)                                                     830
      YB=F(XB,THETA)                                                     840
     *                                                                   850
      FYA=-ABS(XA-(PENTE*YA)-ORD)                                        860
      FYB=-ABS(XB-(PENTE*YB)-ORD)                                        870
     *                                                                   880
   77 IF(FYA-FYB) 80,80,78                                               883
   78 G=XA                                                              900
      XA=XB                                                             910
      XB=G+D-XA                                                         920
      YA=YB                                                             930
      YB=F(XB,THETA)                                                    940
      FYA=FYB                                                           950
      FYB=-ABS(XB-(PENTE*YB)-ORD)                                       960
      GOTO 82                                                           970
     *                                                                  980
   80 D=XB                                                              990
      XB=XA                                                            1000
      XA=G+D-XB                                                        1010
      YB=YA                                                            1020
      YA=F(XA,THETA)                                                   1030
      FYB=FYA                                                          1040
      FYA=-ABS(XA-(PENTE*YA)-ORD)                                      1050
     *                                                                 1060
   82 IF(D-G-ACCUR) 83,83,77                                           1070
     *                                                                 1080
   83 PTMAX=XB                                                         1090
      XMAX=-FYB                                                        1100
      IF(FYB.LT.FYA) RETURN                                            1110
      PTMAX=XA                                                         1120
      XMAX=-FYA                                                        1130
      RETURN                                                           1140
      END                                                              1150
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 GSM

VARIABLES	SN	TYPE	RELOCATION					
172	ACCUR	REAL		200	D	REAL		
205	FYA	REAL		206	FYB	REAL		
177	G	REAL		0	ORD	REAL	F.P.	
0	PENTE	REAL	F.P.	0	PT	REAL	F.P.	

```
1                REAL FUNCTION F(U,THETA)                                    1151
                 F=1000.0*EXP(-THETA*U)                                      1152
                 RETURN                                                      1153
                 END                                                         1154
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
5 F

VARIABLES	SN	TYPE	RELOCATION				
23	F	REAL					
0	U	REAL	F.P.	0	THETA	REAL	F.P.

EXTERNALS	TYPE	ARGS
EXP	REAL	1 LIBRARY

STATISTICS
PROGRAM LENGTH 24B 20

REMERCIEMENTS

Je tiens à remercier Monsieur Sang Nguyen, mon directeur de recherches, qui a su diriger mes travaux avec patience, et sans l'aide duquel ce mémoire n'aurait pu être complété.

Je désire également remercier Messieurs Patrice Marcotte (qui a suggéré l'algorithme LIN1, section 1.2.2.1.) et André Babin pour les nombreuses discussions fructueuses que nous avons eues.

Le conseil national de recherches du Canada, le ministère de l'éducation du Québec ainsi que le centre de recherche et de développement de Transport Canada m'ont accordé une aide financière très appréciée tout au long de mes études de maîtrise.

Je désire également remercier Madame Madeleine Constantin pour son excellent travail de dactylographie.

BIBLIOGRAPHIE

[1] Abadie, J. et Carpentier, J.,"Generalization of the Wolfe reduced gradient method to the case of non-linear constraints", in Optimization, R. Fletcher, editor, Academic Press (1969).

[2] Beckmann, M.J., McGuire, C.R., Winsten, C.B., Studies in the Economics of Transportation, Yale University Press (1956).

[3] Dijkstra, E.W., A note on two problems in connection with graphs, Numerische Mathematik, Vol. 1, (1959).

[4] Florian, M. et Nguyen, S., A method for computing network equilibrium with elastic demands, Transportation Science, Vol. 8, No. 4, (1974).

[5] Fox, B.L., Data structures and computer science techniques in operations research. Département d'informatique et de recherche opérationnelle, Université de Montréal, Publication # 245, (1976).

[6] Geoffrion, A.M., Generalized Benders decomposition, Journal of Optimization Theory and Applications, Vol. 10, No. 4, (1972).

[7] Geoffrion, A.M. Objective function approximations in mathematical programming. Mathematical programming, Vol. 13, No. 1, (1977).

[8] Glover, F., Karney, D., et Klingman, D., The augmented predecessor index method for locating stepping stone paths and assigning dual prices in distribution problems, Transportation Science, Vol. 6, No. 1, (1972).

[9] Knuth, D.E., The Art of Computer Programming, Vol. 1 (second edition) Addison-Wesley, (1973).

[10] Knuth, D.E. The Art of Computer Programming, Vol. 3, Addison-Wesley, (1973).

[11] Lasdon, L.S., Optimization Theory for Large Systems, MacMillan, (1970).

[12] Luenberger, D.G., Introduction to Linear and Non-Linear Programming Addison-Wesley (1973).

[13] Mangasarian, O.L. Non-Linear Programming, McGraw-Hill, (1969).

[14] Nguyen, S., Une approche unifiée des méthodes d'équilibre pour l'affectation du trafic. Département d'informatique et de recherche opérationnelle, Université de Montréal, publication # 171, (1974).

[15] Nguyen, S., Estimating an O/D matrix from network data: a network equilibrium approach. Centre de recherche sur les transports, Université de Montréal, Publication # 60, (1977).

[16] Potts, R.B. et Oliver, R.M. Flows in Transportation Networks. Academic Press, (1972).

[17] Zangwill, W.I., Nonlinear Programming: A Unified Approach. Prentice-Hall (1969).

[18] Zoutendijk, G., Methods of Feasible Directions, Elsevier, (1960).

[19] A propotype Analysis: Search and Choice in Transport Systems Planning, Vol. 2, M.I.T., Department of Civil Engineering, (1968).